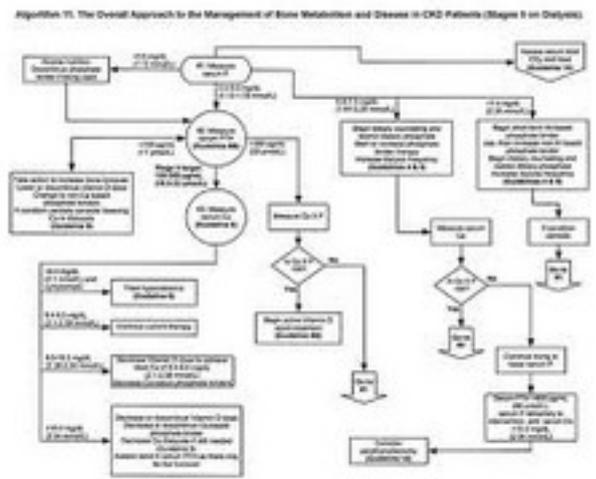


"Languages come and go, but algorithms stand the test of time"

"An algorithm must be seen to be believed."

Donald Knuth

related searches: [simple algorithm](#) [funny algorithm](#) [algorithm flowchart](#) [computer algorithm](#) [algorithm logo](#)



Given the time the entry was posted A and the time of 7:46:43 a.m. December 8, 2005 B , we have t , as their difference in seconds

$$t = A - B$$

and x as the difference between the number of up votes U and the number of down votes D

$$x = U - D$$

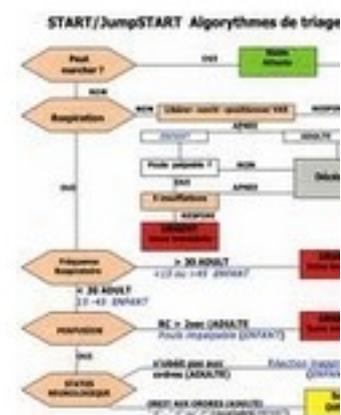
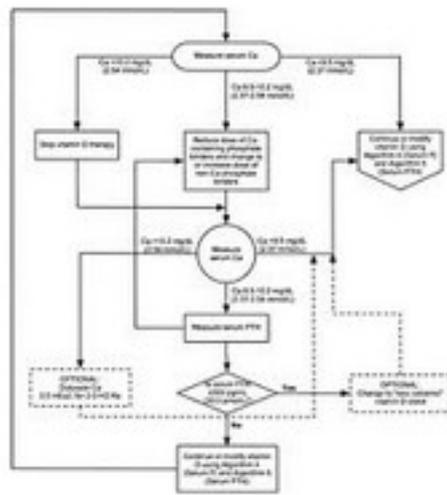
where $y \in (-1, 0, 1)$

$$y = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

and z as the maximal value, of the absolute value of x and 1

$$z = \begin{cases} |x| & \text{if } |x| \geq 1 \\ 1 & \text{if } |x| < 1 \end{cases}$$

we have the rating as a function $f(t, y, z)$

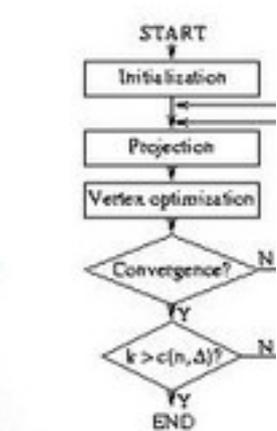
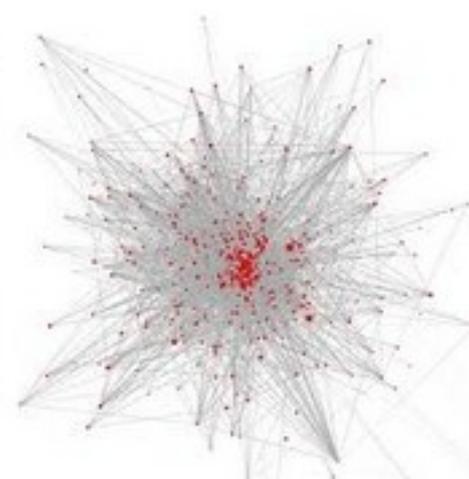
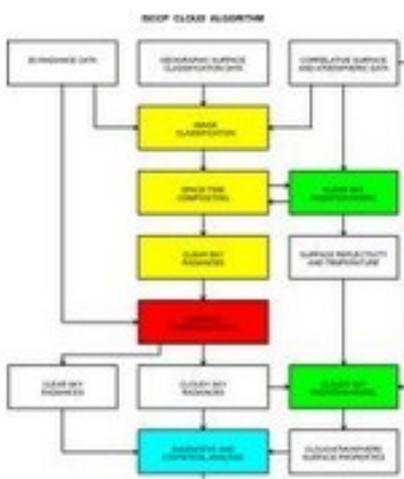
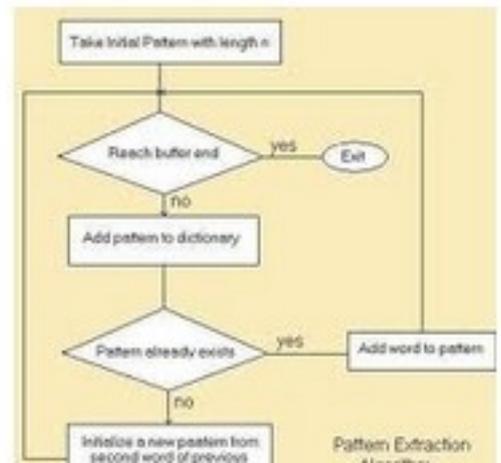
$$f(t, y, z) = \log_{10} z + \frac{yt}{45000}$$


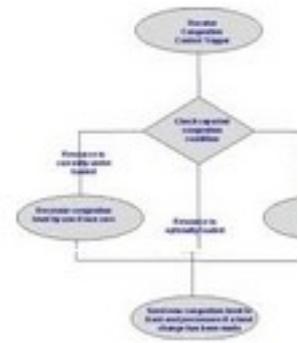
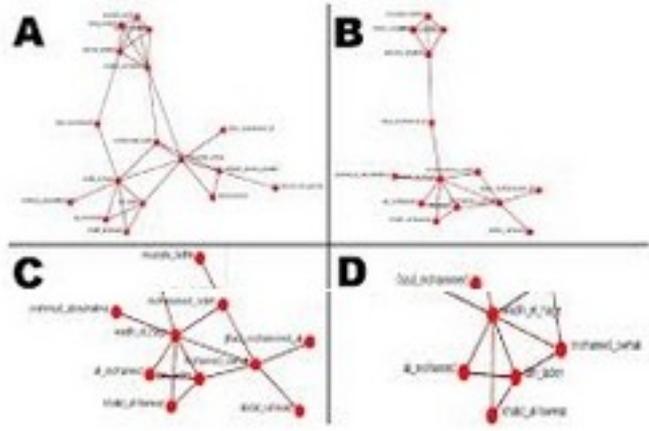
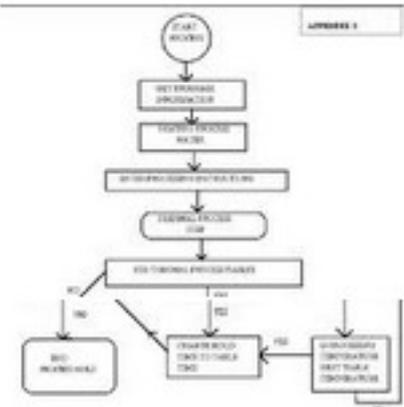
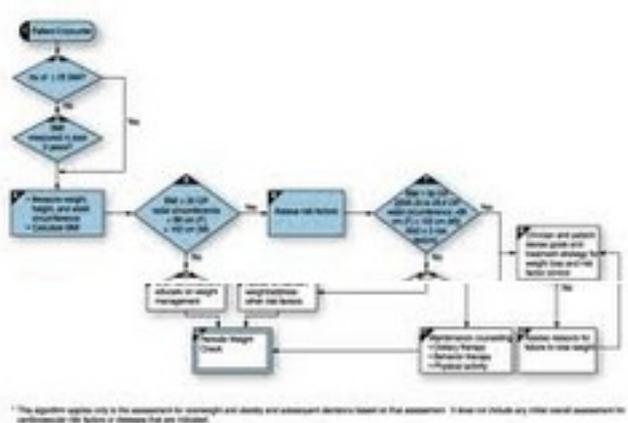
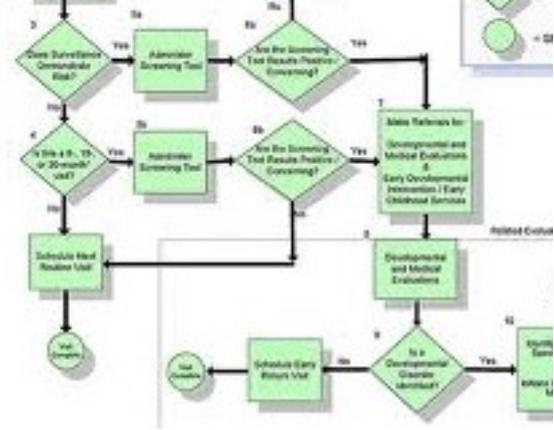
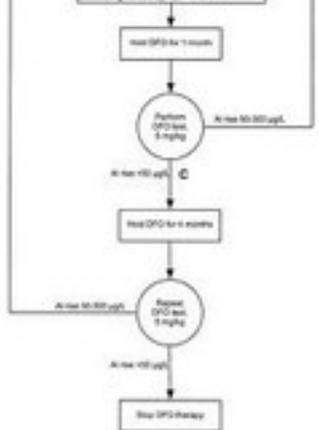
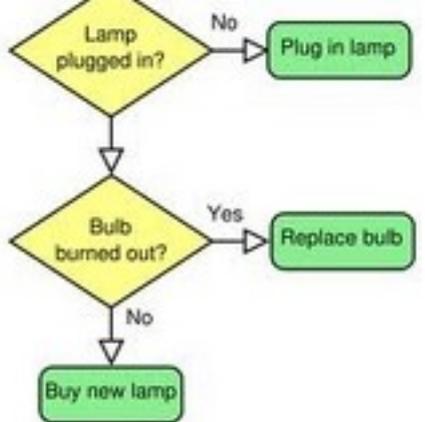
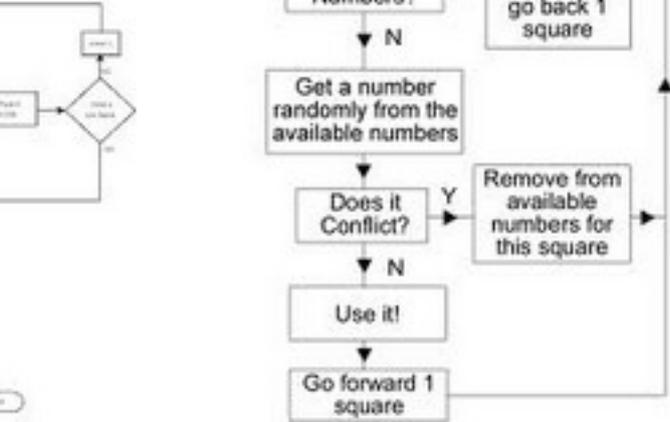
Let $g, L(KAB, n)$ be the user's query $KAB = \{(t, u(t), L(A))\}$ where t is a term, $u(t)$ its normalized frequency in the KAB, $L(A) = \{(k, u(t, k))\}$ is a lexical affinity (t, k) is a lexical frequency in the KAB, $w(t, k)$ its normalized frequency in the KAB, the maximum number of terms for completion.

Let $t \in g$, retrieve $t, u(t), L(A)$ from $L(KAB)$ as a graph $G = (V, E)$ where: $V = \{t \in g \text{ associated with } u(t)\}$ $E = \{(k, t) \text{ associated with } w(t, k) | (t, k) \in V\}$ $V_1, w(t, k) \in L(A)$

Let the connected components of $G, C_1, \dots, C_n, n \geq 0$

Let component $C_i = \{t_1, \dots, t_k\}$ $C_i(t) = 1/k \sum_{t_j \in C_i} w(t, t_j)$ $w = u(C_i)$ $L(A(C_i)) = \{(k, w(t, k))\}$ where: $t \in C_i \cap L(A)$ $w(t) = 1/k \sum_{t_j \in C_i} w(t, t_j)$ $L(A(C_i))$ is decreasing order according to $w(t, k)$

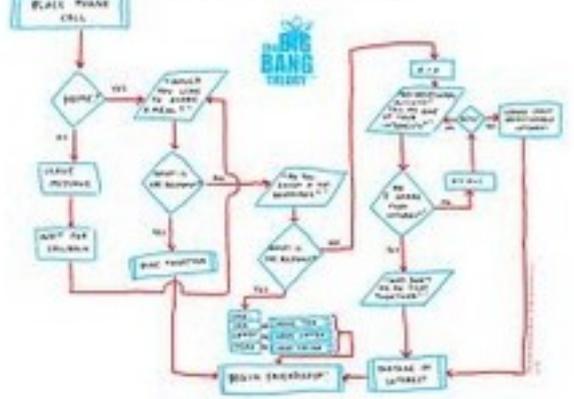




Algorithm used to give change.
Amount owed: 41 cents.

THE FRIENDSHIP ALGORITHM

DR. SHELDON COOPER, Ph.D



Recipe
CHOCOLATE CAKE

4 oz. chocolate
1 cup butter
2 cups sugar

3 eggs
1 tsp vanilla
1 cup flour

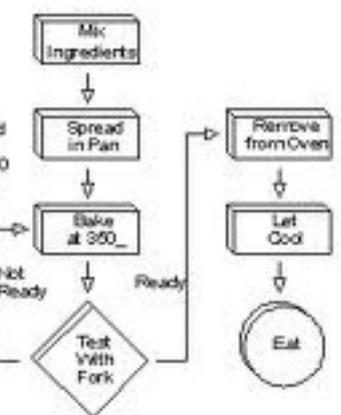
Melt chocolate and butter. Stir sugar into melted chocolate. Stir in eggs and vanilla. Mix in flour. Spread mix in greased pan. Bake at 350, for 40 minutes or until inserted fork comes out almost clean. Cool in pan before eating.

Program Code

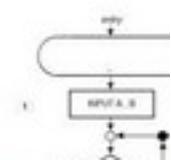
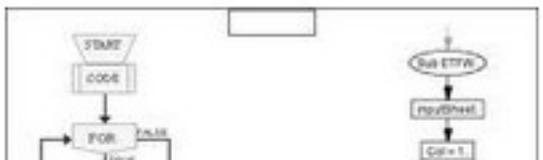
```

Declare variables:
chocolate  eggs    mix
butter     vanilla  flour
sugar

mix = melted ((4*chocolate) + butter)
mix = stir (mix + (2*sugar))
mix = stir (mix + (3*eggs) + vanilla)
mix = mix + flour
spread (mix)
while not clean (fork)
  bake (mix, 350)
  
```

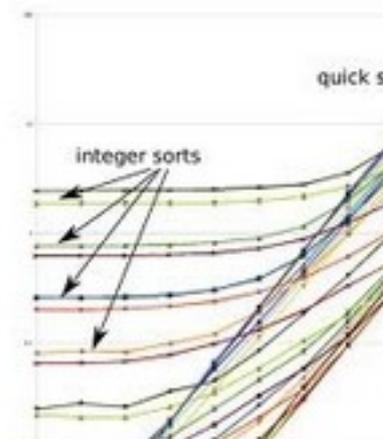
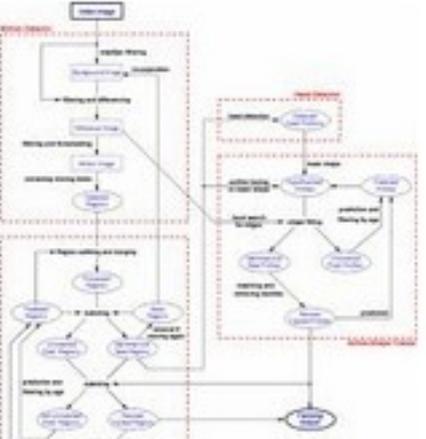
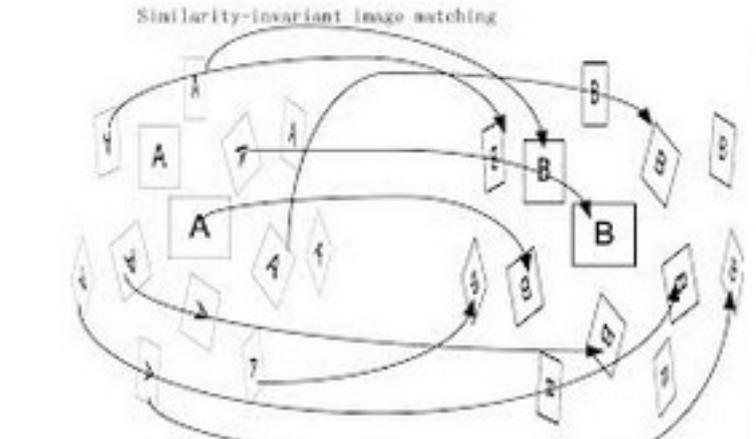
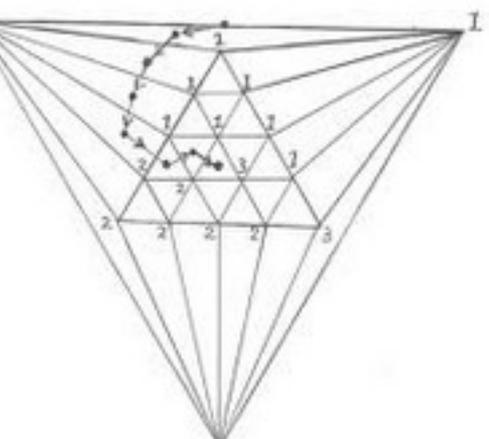
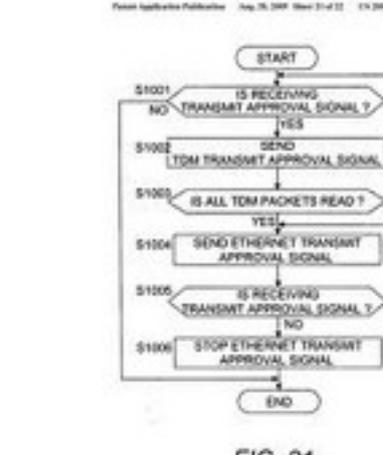
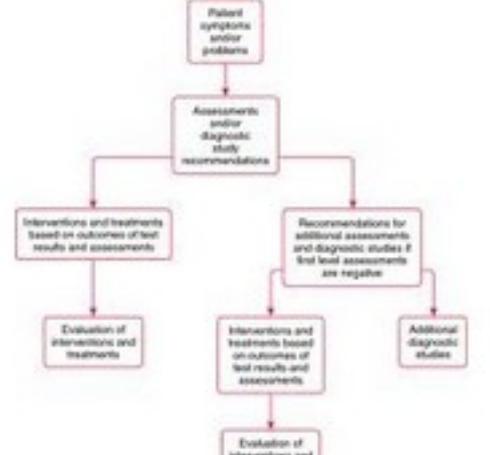
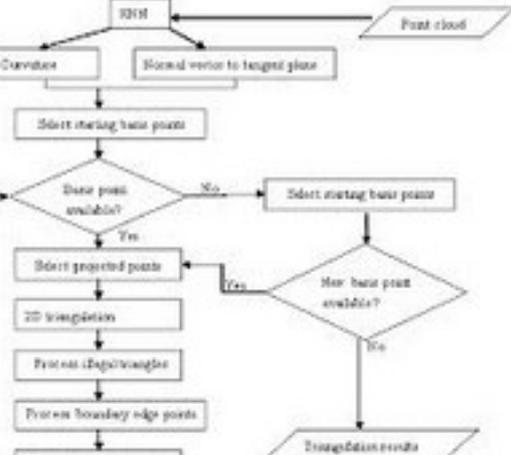
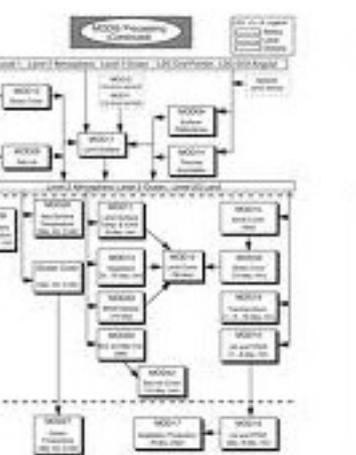
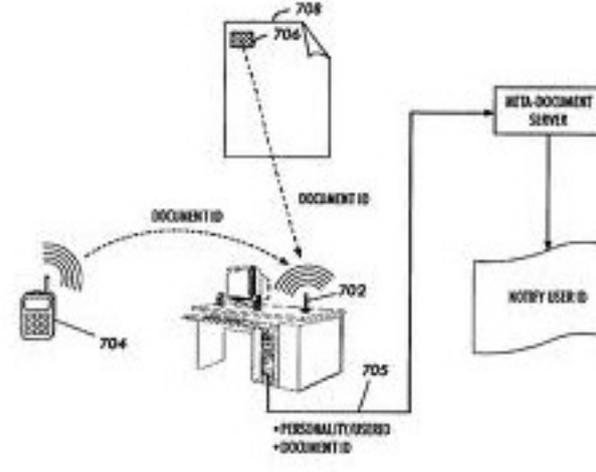
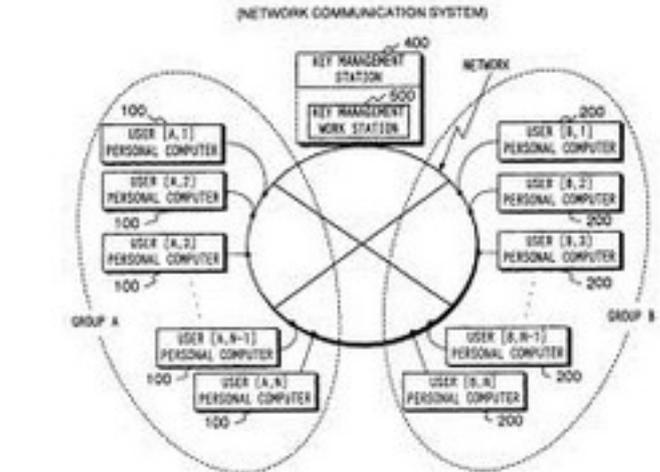
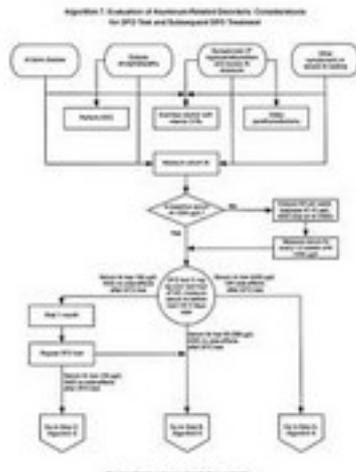


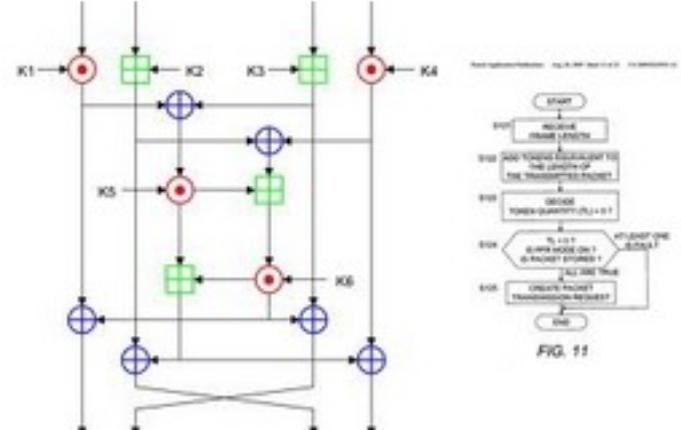
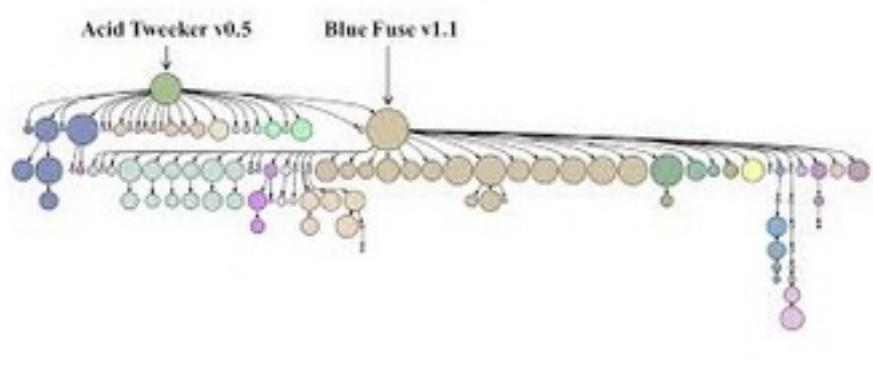
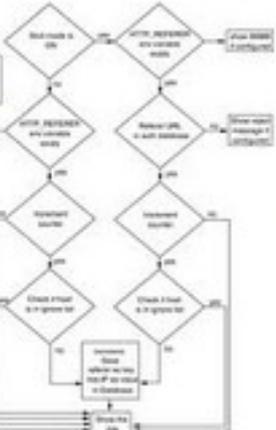
$$\frac{1}{2} \log(n) - \frac{\sum \Lambda(d)}{d} + T(n)$$



The Search Algorithm

The search algorithm is a process of finding information or data. It involves a series of steps and decisions to locate the desired information. The algorithm starts with a search space and narrows it down through various operations like filtering, sorting, and indexing. The process continues until the target information is found or the search space is exhausted. The efficiency of the search algorithm depends on the complexity of the search space and the quality of the search strategy used.





Greedy Algorithm

A greedy algorithm obtains an optimal sequence of choices. At each decision point, it chooses the locally optimal choice. In other words, it makes the choice that seems best at the moment, without considering the consequences of that choice.

Problems solved optimally by greedy algorithms are those that have an optimal substructure. In other words, the optimal solution to the problem contains within it the optimal solution to a subproblem.

Today we look at two variations of the greedy algorithm: Huffman coding and Kruskal's algorithm for finding a minimum spanning tree.

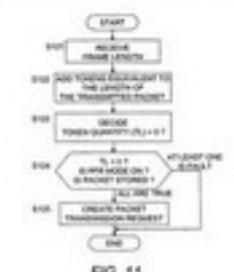
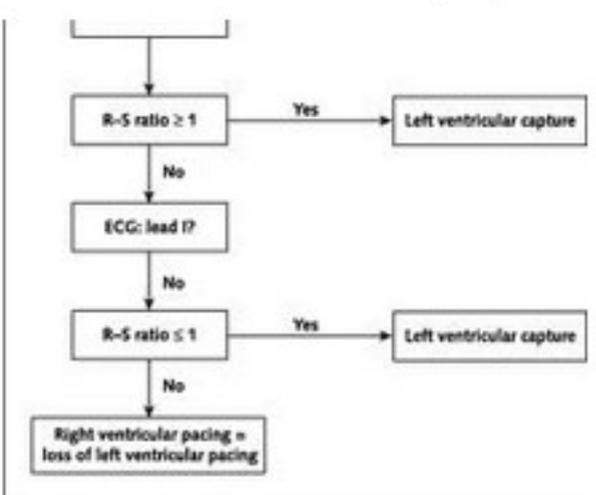
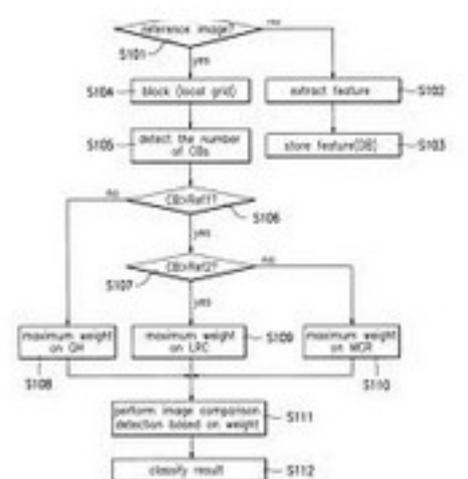


FIG. 11



Given two numbers not prime is one another, to find their greatest common measure.

Let AB, CD be the two given numbers not prime to one another.

That it is required to find the greatest common measure of AB, CD .

If now CD measures AB —and it also measures itself— CD is a common measure of AB, CD .

And it is manifest that it is also the greatest; for no greater number will be left which will measure CD .

But, if CD does not measure AB , then, the line of the numbers AB, CD being continually subtracted from the greater, some number will be left which will measure the one before it.

For an unit will not be left, otherwise AB, CD will be prime to one another [vii. 1], which is contrary to the hypothesis.

Therefore some number will be left which will measure the one before it.

Now let CD measure AE , leave EA less than itself, let EA measure DF , leave FD less than itself, and let CF measure AE .

Since then, CF measures AE , and AE measures DF , therefore CF will also measure DF .

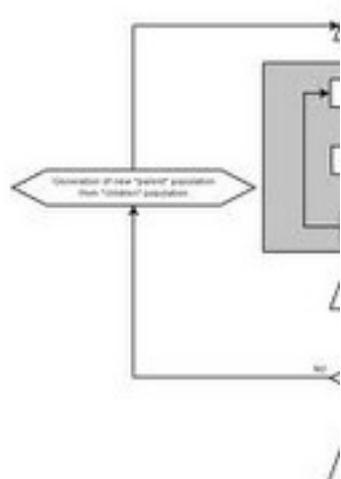
But it also measures itself; therefore it will also measure the whole CD .

But CD measures AE ; therefore CF also measures AE .

But it also measures EA ; therefore it will also measure the whole AB .

But it also measures CD ; therefore CF measures AB, CD .

Therefore CF is a common measure of AB, CD .



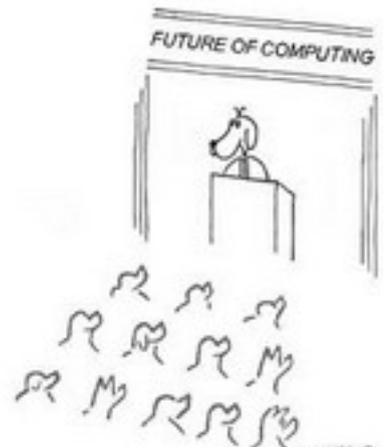
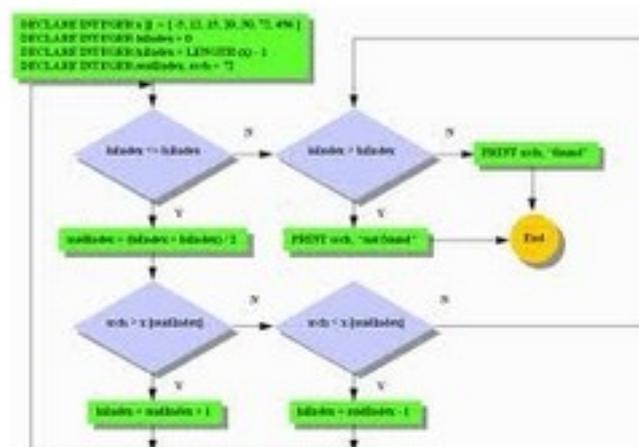
$\in R$.

$\in R$ the gcd of a and b together with $s, t \in R$ that $g = sa + tb$.

$r_0 := \text{normal}(a); s_0 := a_0^{-1}; t_0 := 0$
 $r_1 := \text{normal}(b); s_1 := 0; t_1 := a_1^{-1}$

$\neq 0$ repeat

-2 QRO r_{i-1}
 -2 QRM r_{i-1}
 (r_i)
 $\text{normal}(r_i)$
 -2 $q_i = \text{floor}(r_{i-1}/r_i)$
 -2 $s_i = s_{i-1} - q_i t_{i-1}$
 -2 $t_i = t_{i-1} - q_i s_{i-1}$
 1
 -2 $s_i = s_{i-2} t_{i-2}$



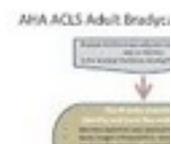
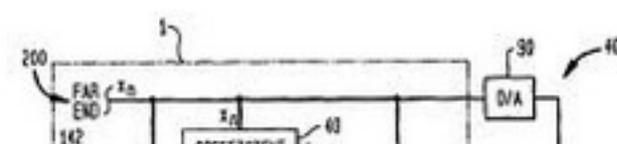
Get Fit Using These Simple And Easy Methods

Posted on April 5, 2012 by Tony

If you agree you are too active to get time and energy to exercise, you want available a fantastic exercise routine a lot sooner than you imagine, here are numerous ideas that could show you to improve your workout in a way which enable it to pay day loan you stay healthy and keep the kitchen clean.

To assist you to recover loan coming from a tricky exercise routine, you should exercise the next day. You want to do this softly, as that one could elevate on one occasion. Try to do 25 repetitions in this, you'll have additional blood and nutrients sent to the muscle.

Climbing is a terrific way to stay fit while not having to expend even a penny. Circumstances car park is a superb destination for a walk. You have effectively groomed, predesignated hiking trails. You won't just have cardiovascular exercise routine, there is however a high probability of taking in some stunning views.



Dictionary of Algorithms and Data Structures

This web site is hosted in part by the [Software and Systems Division](#), [Information Technology Laboratory](#).

This is a dictionary of algorithms, algorithmic techniques, data structures, archetypal problems, and related definitions. Algorithms include common functions, such as [Ackermann's function](#). Problems include [traveling salesman](#) and [Byzantine generals](#). Some entries have links to [implementations](#) and more information. Index pages list entries by [area](#) and by [type](#). The [two-level index](#) has a total download 1/20 as big as this page.

Don't use this site to cheat.

To define or correct terms, please contact [Paul E. Black](#). We do not include algorithms particular to business data processing, communications, operating systems or distributed algorithms, programming languages, AI, graphics, or numerical analysis: it is tough enough covering "general" algorithms and data structures.

Some terms with a leading variable, such as n -way, m -dimensional, or p -branching, are under [k](#). You may find useful entries in [A Glossary of Computer Oriented Abbreviations and Acronyms](#).

DADS is Moving



After more than a decade of service, I will end my editing of DADS. Everything will move to the [FASTAR group](#) at University of Pretoria (South Africa), Stellenbosch University (South Africa), and Eindhoven University (Netherlands).

To look up words or phrases, enter them in the box, then click the button.



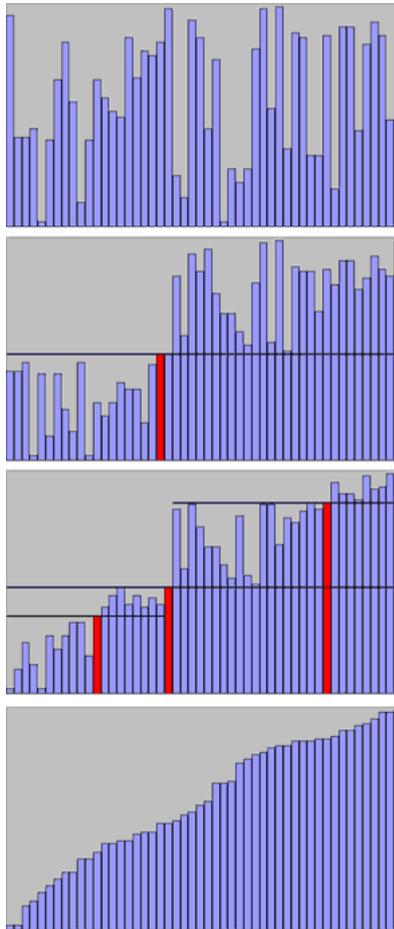
Web DADS

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#)

<http://xlinux.nist.gov/dads/>

<http://xlinux.nist.gov/dads//HTML/quicksort.html>

Algorithme de tri rapide



http://fr.wikipedia.org/wiki/Tri_rapide

Programmer pour l'intelligence collective



Qu'est-ce qui peut bien différencier des applications Web 2.0 comme Facebook, Google, Amazon ou eBay de celles qui échouent à attirer des millions d'utilisateurs ? L'innovation ? L'utilité ? L'effet de mode ? La réponse est : les algorithmes d'exploitation de l'intelligence collective !

En effet, sans algorithme capable de capter les comportements des internautes et de répondre à leurs besoins spécifiques, une application Web 2.0 n'est qu'une coquille vide. Programmer pour l'intelligence collective vous fait entrer dans les coulisses des applications qui marquent leur temps en démontant les mécanismes des algorithmes qui les rendent si efficaces. Dans cet ouvrage, vous apprendrez entre autres à :

- Mettre au point des filtrages collaboratifs et des systèmes de recommandations de type del.icio.us ;
- Révéler l'existence de groupes à la manière des groupes de consommateurs d'Amazon ;
- Créer des algorithmes de recherche, d'indexation et de classement comme ceux de Google ;
- Filtrer tous types de documents à la manière des anti-spams ;
- Décrypter des mécanismes décisionnels avec la modélisation en arbres ;
- Établir des modèles de prix dans l'esprit de l'API eBay ;
- Programmer une intelligence évolutive à l'aide d'algorithmes génétiques.

((1, 327, 23), (1, 327, 162), (1, 327, 243), (1, 327, 261),
(1, 327, 269), (1, 327, 436), (1, 327, 953),...

Vous noterez que chaque identifiant d'URL est retourné plusieurs fois pour différentes combinaisons de positions de mots. Les sections suivantes présenteront plusieurs manières de classer les résultats. Le *classement basé sur le contenu* s'appuie sur plusieurs métriques possibles avec uniquement le contenu de la page pour déterminer la pertinence de la requête. Le *classement basé sur les liens entrants* se fonde sur la structure des liens du site pour déterminer les aspects importants. Pour améliorer le classement au fil du temps, nous verrons également un système permettant de connaître les liens activés par les utilisateurs suite à leurs recherches.

Classement basé sur le contenu

Pour le moment, nous réussissons à obtenir les pages qui correspondent à des requêtes, mais les résultats sont donnés dans l'ordre de l'exploration. Si le jeu de pages est volumineux, nous allons passer beaucoup de temps à examiner une grande quantité de contenu non pertinent pour chaque occurrence des termes de la requête afin de ne garder que les pages qui nous concernent réellement. Pour résoudre ce problème, nous devons définir des mécanismes permettant de donner un *score* aux pages pour une requête donnée et de retourner les pages en commençant par celles qui ont les meilleurs scores.

Cette section présente plusieurs métriques de calcul d'un score uniquement en fonction de la requête et du contenu de la page :

Fréquence des mots

Le nombre d'occurrences des mots de la requête dans le document peut aider à déterminer la pertinence du document.

Position dans le document

Le thème principal d'un document apparaîtra probablement vers le début.

Distance des mots

Si la requête contient plusieurs mots, ils doivent être proches les uns des autres dans le document.

Les premiers moteurs de recherche employaient souvent ces métriques et produisaient des résultats intéressants. Les sections suivantes s'attacheront à améliorer les résultats en

La fonction de normalisation prend en arguments un dictionnaire d'identifiants et un dictionnaire de scores et retourne un nouveau dictionnaire constitué des mêmes identifiants, mais avec des scores entre 0 et 1. Chaque score est ajusté selon sa distance avec le meilleur résultat, qui a toujours un score final égal à 1. Pour utiliser cette fonction, vous devez simplement lui passer une liste de scores et indiquer si le meilleur est représenté par la valeur la plus grande ou la plus petite :

```
def normaliserscores(self, scores, petitEstMeilleur=0):
    petiteval=0.00001 # Éviter une division par zéro.
    if petitEstMeilleur:
        scoremin=min(scores.values())
        return dict([(u,float(scoremin)/max(petiteval,1)) for (u,l) \
                    in scores.items()])
    else:
        scoremax=max(scores.values())
        if scoremax==0: scoremax=petiteval
        return dict([(u,float(c)/scoremax) for (u,c) in scores.items()])
```

Toutes les fonctions de calcul de scores appellent celle-ci afin de normaliser ses résultats et obtenir une valeur dans l'intervalle 0 à 1.

Fréquence des mots

Avec la métrique basée sur la fréquence des mots, le score d'une page est déterminé par le nombre d'occurrences des mots de la requête dans la page. Si nous recherchons « python », nous souhaitons obtenir une page concernant Python (ou les pythons) qui contient plusieurs fois ce mot et non une page sur un musicien qui mentionne une fois que son animal domestique est un python.

Voici la fonction de calcul des scores basé sur la fréquence des mots. Ajoutez-la à la classe chercheur :

```
def scorefrequence(self, lignes):
    valeurs=dict([(ligne[0],0) for ligne in lignes])
    for ligne in lignes: valeurs[ligne[0]]+=1
    return self.normaliserscores(valeurs)
```

Elle crée un dictionnaire avec une entrée pour chaque identifiant d'URL unique de lignes et comptabilise le nombre d'occurrences de chaque élément. Ensuite, elle normalise les scores (dans ce cas, le meilleur score est celui le plus élevé) et retourne le résultat.

Pour activer cette méthode de calcul, modifiez de la manière suivante la ligne lesponds dans la fonction obtenirlistenotes :

```
0.062310      http://kiwitobes.com/wiki/
0.043976      http://kiwitobes.com/wiki/
0.036394      http://kiwitobes.com/wiki/
...
```

La page concernant la programmation fonctionne en premier, suivie d'autres pages pertinentes. V est quatre fois supérieur à celui de la page placée à la fin. Les moteurs de recherche n'indiquent pas les scores à moins qu'ils soient très utiles à certaines applications. Par exemple, nous pouvons indiquer au utilisateur sur la page ayant le meilleur score de toujours afficher les résultats dans une taille de police proportionnelle à ce score.

Position dans le document

Une autre métrique simple pour déterminer la pertinence d'un terme consiste à examiner l'emplacement d'un terme dans le document. Par exemple, le terme correspondant apparaît dans le titre. Pour exploiter cette caractéristique, nous pouvons donner un score plus élevé lorsque le terme de la requête apparaît dans le titre. Lors de l'indexation des pages, nous avons pu constater que les mots et le titre de la page se trouve au début de la page.

Ajoutez la méthode suivante à la classe chercheur :

```
def scoreposition(self, lignes):
    positions=dict([(ligne[0],1000000) for ligne in lignes])
    for ligne in lignes:
        pos=sum(ligne[1:])
        if pos<positions[ligne[0]]: positions[ligne[0]]=pos
    return self.normaliserscores(positions)
```

Le premier élément de chaque ligne est l'identifiant de la page et les autres éléments sont les différents termes recherchés. Chaque identifiant est associé à une liste de positions pour chaque combinaison de positions. Pour chaque ligne, nous calculons la somme des positions de tous les mots et compare ce résultat avec la position du terme pour l'URL. Elle passe les résultats finaux à la fonction normaliserscores. petitEstMeilleur=1 signifie que l'URL dont la position est la plus basse obtient le score 1,0.

Pour voir les résultats obtenus uniquement avec ce score, nous pouvons modifier la fonction obtenirlistenotes :

« Haskell is a standardized pure functional programming language ») conduit à un meilleur score.

Il est important de comprendre qu'aucune des métriques décrites jusqu'alors n'est meilleure que l'autre dans tous les cas. Les deux listes produites sont pertinentes selon l'objectif de la recherche. Différentes combinaisons de poids sont nécessaires pour obtenir les bons résultats, selon les documents traités et les objectifs des applications. Vous pouvez expérimenter différents poids pour les deux métriques en modifiant la ligne des poids de la manière suivante :

```
lespoids=[(1.0,self.scorefrequence(lignes)),  
          (1.5,self.scoreposition(lignes))]
```

Testez différents poids et différentes requêtes pour en voir l'impact sur les résultats.

La position est une métrique plus difficile à tromper que la fréquence des mots, car les auteurs des pages ne peuvent placer qu'un seul mot en premier dans leurs documents et la répétition n'a aucun impact sur les résultats.

Distance des mots

Quand une requête concerne plusieurs mots, il est souvent utile de rechercher les résultats dans lesquels ces mots sont proches les uns des autres dans la page. En effet, lorsque les utilisateurs effectuent des requêtes avec plusieurs mots, ils s'intéressent aux pages qui contiennent conceptuellement ces mots. Cette approche est un peu plus souple que les requêtes de phrases entre guillemets acceptées par la plupart des moteurs de recherche. En effet, les mots doivent alors apparaître dans l'ordre correct sans autre mot supplémentaire. Dans notre solution, la métrique tolère un ordre différent et la présence de mots autres que ceux de la requête.

La fonction `scoredistance` ressemble à `scoreposition` :

```
def scoredistance(self,lignes):  
    # S'il n'y a qu'un seul mot, tout le monde gagne !  
    if len(lignes[0])<=2: return dict([(ligne[0],1.0) for ligne in lignes])  
  
    # Initialiser le dictionnaire avec les valeurs les plus grandes.  
    distancemin=dict([(ligne[0],1000000) for ligne in lignes])  
  
    for ligne in lignes:  
        dist=sum([abs(ligne[i]-ligne[i-1]) for i in range(2,len(ligne))])  
        if dist<distancemin[ligne[0]]: distancemin[ligne[0]]=dist  
    return self.normaliserscores(distancemin,petitEstMeilleur=1)
```

La principale différence se trouve dans le parcours des positions (la ligne en gras). Là où nous calculons la différence entre chaque position et la position précédente. Puisque nous testons toutes les combinaisons de distances sont retournées par la requête, nous sommes con-

Utiliser les liens entrants

Les métriques de scores présentées jusqu'à présent s'appuient sur le contenu de la page. Bien que de nombreux moteurs de recherche fonctionnent de cette manière, les résultats peuvent souvent être améliorés en utilisant des liens entrants que d'autres fournissent sur la page, en particulier ceux qui ont des commentaires. Cette approche se révèle particulièrement utile pour l'indexation des pages à la valeur incertaine ou des pages qui n'ont pas de liens entrants. Il y a peu de chances que d'autres pages les référencent.

Le robot développé au début de ce chapitre collecte déjà toutes les pages et il est donc inutile de le modifier. La seule modification à apporter est d'ajouter des liens entrants d'URL de la source et de la cible pour chaque lien entrant. Cela connecte les mots aux liens.

Compteur simple

Le traitement le plus simple des liens entrants consiste à compter le nombre de liens entrants sur chaque page et à prendre le nombre total de liens entrants comme métrique. Les pages universitaires sont souvent classées de cette manière, leur impact est mesuré par le nombre d'autres articles qui les référencent. La fonction de calcul du dictionnaire de compteurs en consultant la table des liens par URL dans `lignes`. Elle retourne ensuite les scores normalisés.

```
def scoreliensentrants(self,lignes):  
    urluniques=set([ligne[0] for ligne in lignes])  
    compteurentrant=dict([(u,self.con.execute('select count(*) from liens where iddst=%d and idsrc=%d' % (u,v)) for u in urluniques)])  
    return self.normaliserscores(compteurentrant)
```

Bien évidemment, l'emploi de cette métrique seule retourne des résultats qui ne sont pas toujours pertinents. Les pages contenant les termes recherchés, classées uniquement par le nombre de liens entrants existants. Dans notre jeu de données, « Python » a plus de liens entrants que « Python », mais nous souhaitons que « Python » soit plus pertinent dans les résultats si c'est le thème qui nous intéresse. Pour améliorer le classement, nous devons associer la métrique des liens entrants à la métrique des liens sortants précédemment.

Cet algorithme donne un poids équivalent à chaque lien entrant. Cela ouvre la porte aux manipulations car il est facile de modifier le poids de chaque lien entrant vers une page afin d'en augmenter le score. Il est également possible de modifier le poids de chaque lien entrant pour que les pages les plus populaires soient plus intéressées par les résultats qui ont un lien entrant. Nous allons voir comment faire en sorte que les pages les plus populaires aient plus d'impact sur le classement.

Cet algorithme attribue à chaque page un score qui indique son importance. Cette valeur est calculée d'après l'importance de toutes les autres pages qui la référencent et le nombre de liens présents dans les autres pages.



En théorie, PageRank (dont le nom est lié à l'un de ses créateurs, Larry Page) calcule la probabilité qu'une personne cliquant aléatoirement sur des liens arrive à une certaine page. Plus les liens vers la page proviennent de pages populaires, plus cette personne a de chances d'y arriver par hasard. Bien entendu, si l'utilisateur clique indéfiniment, il finira pas atteindre chaque page, mais la majorité des internautes s'arrête de surfer après un certain temps. Pour mettre en œuvre cette notion, PageRank utilise également un *facteur d'amortissement* égal à 0,85. Il signifie qu'il existe 85 % de chances qu'un utilisateur continue à cliquer sur des liens à partir d'une page donnée.

Figure 4-3 montre un exemple de pages et de liens.

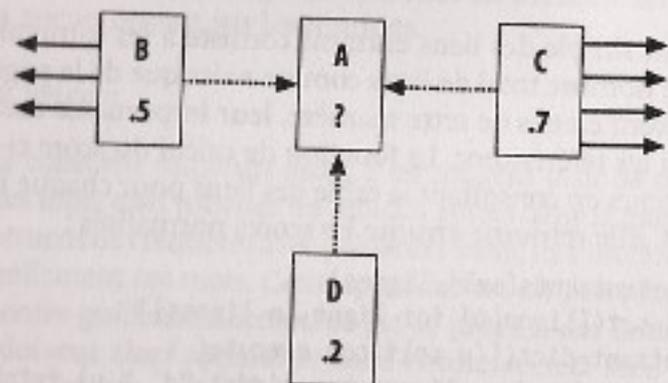


Figure 4-3. Calculer le PageRank de A

Pages B, C et D font toutes référence à la page A et leur PageRank a déjà été calculé. La page B possède également des liens vers trois autres pages et la page C vers quatre pages. La page D contient uniquement des liens vers A. Pour obtenir le PageRank nous prenons le PageRank (PR) de chaque page qui référence A et le divisons par le nombre total de liens de la page, puis multiplions cette valeur par un facteur d'amortissement égal à 0,85, et ajoutons au résultat la valeur minimum 0,15. Voici le calcul de

$$\begin{aligned}
 &= 0,15 + 0,85 * (PR(B)/liens(B) + PR(C)/liens(C) + PR(D)/liens(D)) \\
 &= 0,15 + 0,85 * (0,5/4 + 0,7/5 + 0,2/1) \\
 &= 0,15 + 0,85 * (0,125 + 0,14 + 0,2) \\
 &= 0,15 + 0,85 * 0,465 \\
 &= 0,54525
 \end{aligned}$$

connus. De même, il est impossible de calculer leur score sans les pages qui les référencent. Comment peut-on donc calculer le PageRank d'un ensemble de pages qui n'ont pas encore leur PageRank ?

La solution consiste à fixer tous les PageRank à une valeur initiale de 1,0, mais la valeur choisie ne fait aucune différence dans les calculs. Après chaque itération, le PageRank de chaque page est recalculé à partir de son PageRank réel. Le nombre d'itérations nécessaires varie en fonction du nombre de pages, mais, avec notre petit jeu, 20 devraient suffire.

Puisque le calcul du PageRank prend du temps et reste le même, nous créons une fonction qui le précalcule pour chaque page et stocke la valeur dans une table. Cette fonction recalculera tous les PageRank à chaque fois que vous ajoutez une page. Ajoutez la fonction suivante à la classe robot :

```

def calculerpagerank(self, iterations=20):
    # Supprimer la table des PageRank actuels.
    self.con.execute('drop table if exists pagerank')
    self.con.execute('create table pagerank(idurl text, score float)')

    # Initialiser chaque url avec un PageRank égal à 1,0
    self.con.execute('insert into pagerank select idurl, 1.0 from urls')
    self.bdvalider()

    for i in range(iterations):
        print "Iteration %d" % (i)
        for (idurl,) in self.con.execute('select rowid, idurl from urls'):
            pr=0.15

            # Parcourir toutes les pages faisant référence à la page
            for (lieur,) in self.con.execute('select idsrc from liens where idsrc=?'):
                # Obtenir le PageRank du lieur.
                prlieur=self.con.execute('select score from pagerank where idurl=?')

            # Obtenir le nombre total de liens dans la page
            nbliens=self.con.execute('select count(*) from liens where idsrc=?')

            pr+=0.85*(prlieur/nbliens)

        self.con.execute('update pagerank set score=%f where idurl=?')
        self.bdvalider()
  
```

Page Rank

PageRank is Google's system for ranking web pages. A page with a higher PageRank is deemed more important and is more likely to be listed above a page with a lower PageRank.

Src: <http://www.googleguide.com/pagerank.html>

Page Rank

Google describes PageRank:

“PageRank relies on the uniquely democratic nature of the web by using its vast link structure as an indicator of an individual page’s value. In essence, Google interprets a link from page A to page B as a vote, by page A, for page B. But, Google looks at more than the sheer volume of votes, or links a page receives; it also analyzes the page that casts the vote. Votes cast by pages that are themselves “important” weigh more heavily and help to make other pages “important.””

Src: <http://www.googleguide.com/pagerank.html>



Page Rank

- Liste de films
- Introduction to PageRank | Google Algorithm | Matt Cutts
<http://www.youtube.com/watch?v=g9p1ji4EFLc>
- Tsila Hassine, Verbindungen-Jonctions 10 (0 → 10 min, Shmoogle)
<http://video.constantvzw.org/vj10/Tsilla-Hassine.ogg>
- Google CEO Eric Schmidt On Google's Secret Ranking Algorithm
<http://www.youtube.com/watch?v=iVf267F-0pE>
-

Élection, algorithme social



http://fr.wikipedia.org/wiki/Syst%C3%A8me_%C3%A9lectoral

Utiliser le contenu des liens

Une autre manière très puissante de classer les recherches consiste à utiliser le texte des liens vers une page pour décider de sa pertinence. Les informations obtenues depuis les liens sont très souvent meilleures que celles obtenues directement depuis la page, car il arrive que les créateurs de sites incluent une courte description des cibles des liens.

La méthode de notation des pages basée sur le contenu des liens attend un argument supplémentaire, qui n'est autre que la liste des identifiants de mots produite lorsque nous effectuons une requête. Ajoutez la méthode suivante à chercheur :

```
def scoretexteliens(self,lignes,idsmots):
    scoreslien=dict([(ligne[0],0) for ligne in lignes])
    for idmot in idsmots:
        cour=self.con.execute(
            'select liens.idsrc,liens.iddst from motsliens,liens \
            where idmot=%d and motsliens.idlien=liens.rowid' % idmot)
        for (idsrc,iddst) in cour:
            if iddst in scoreslien:
                pr=self.con.execute('select score from pagerank where idurl=%d'
                    % idsrc).fetchone()[0]
                scoreslien[iddst]+=pr
        scoremax=max(scoreslien.values())
        scoresnormalises=dict([(u,float(l)/scoremax)
            for (u,l) in scoreslien.items()])
    return scoresnormalises
```

Ce code parcourt tous les mots présents dans idsmots à la recherche de liens qui les contiennent. Si la cible du lien correspond à l'un des résultats de la recherche, le PageRank de la source du lien est alors additionné au score final de la page de destination. Lorsque de nombreuses pages importantes font référence à une page à l'aide de liens contenant les termes de la requête, cette page obtient un score très élevé. Toutes les pages des résultats qui n'ont aucun lien avec les mots indiqués obtiennent un score égal à 0.

Pour activer le classement en fonction du contenu des liens, ajoutez simplement la ligne suivante dans la liste lespoids :

```
(1.0,self.scoretexteliens(lignes,idsmots))
```

Il n'existe aucun ensemble de poids standard pour ces métriques qui fonctionnera dans toutes les situations : même les sites de recherche les plus importants changent fréquemment leurs méthodes de classement des résultats. Les métriques que vous utiliserez et les poids que vous leur donnerez dépendent fortement de l'application que vous développez.

Exploiter les clics

requête et l'exploitation de ces informations pour améliorer le classement des résultats. Pour cela, nous allons construire un *réseau de neurones artificiel* qui apprendra en recevant les mots de la requête, les résultats de la recherche présentés à l'utilisateur et le choix de celui-ci. Lorsque le réseau aura appris à partir de différentes requêtes, nous pourrons l'utiliser pour modifier l'ordre des résultats de la recherche et mieux refléter les clics passés effectués par les utilisateurs.

Concevoir un réseau de suivi des clics

Bien qu'il existe de nombreux types de réseaux neuronaux, ils sont tous constitués d'un ensemble de nœuds (les *neurones*) interconnectés. Le réseau que nous allons construire est du type *perceptron multicouche* (PMC). Un tel réseau est constitué de plusieurs couches de neurones, la première représentant l'entrée — dans ce cas, les mots saisis par l'utilisateur. La dernière couche représente la sortie, qui, dans cet exemple, est une liste de poids pour les différentes URL retournées.

Le nombre de couches intermédiaires n'est pas défini, mais le réseau de cet exemple n'en utilise qu'une. Il s'agit d'une *couche cachée* car le monde extérieur au réseau n'interagit jamais directement avec elle et elle répond aux combinaisons des entrées. Dans notre cas, une combinaison d'entrées est un ensemble de mots, nous pouvons également la voir comme une *couche de requête*. La figure 4-4 présente la structure du réseau. Tous les nœuds de la couche d'entrée sont connectés à ceux de la couche cachée, qui sont tous connectés aux nœuds de la couche de sortie.

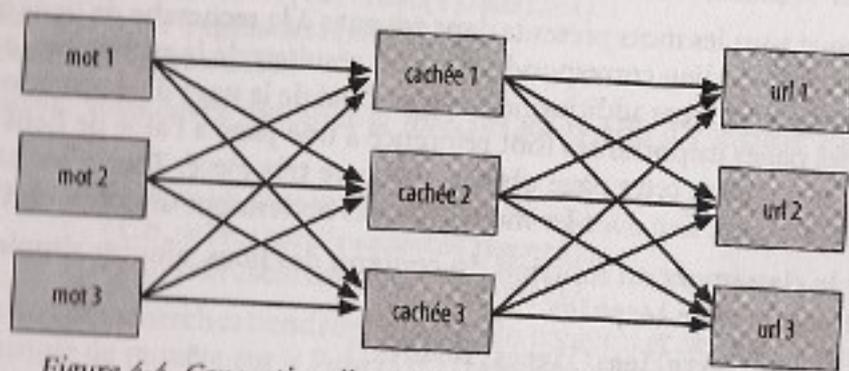


Figure 4-4. Conception d'un réseau neuronal de suivi des clics

Si le réseau de neurones génère les meilleurs résultats d'une requête, les valeurs des nœuds d'entrée pour les mots de cette requête sont égales à 1. Les sorties de ces nœuds sont activées et tentent d'activer la couche cachée. À leur tour, les nœuds de la couche cachée qui reçoivent une entrée suffisamment forte activent leur sortie et tentent d'activer les nœuds de la couche de sortie.

Les nœuds de la couche de sortie deviennent alors actifs à différents degrés et leur niveau d'activité permet de déterminer...

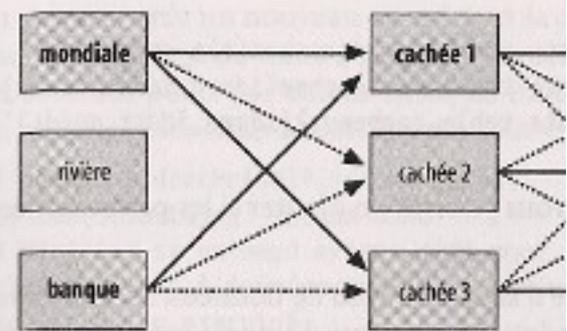


Figure 4-5. Réponse du réseau neuronal à une requête

Bien entendu, tout cela suppose que la solidité des connexions du réseau est le résultat d'un apprentissage du réseau effectué chaque fois que l'utilisateur clique sur un lien des résultats. Dans le réseau illustré, les nœuds ont précédemment cliqué sur le résultat « Banque mondiale » et cela a influencé les poids des nœuds. Cette section explique comment entraîner le réseau à l'aide de l'algorithme de rétro-propagation.

Vous vous demandez peut-être pourquoi il faudrait utiliser un réseau neuronal à la place d'une simple liste de liens. La réponse réside dans le fait qu'il est difficile de faire des associations raisonnables sur les résultats de requêtes qu'il n'y a pas de similitudes avec d'autres requêtes. Par ailleurs, les résultats de recherche doivent faire face à un grand nombre d'applications et doivent faire face à des solutions pour l'intelligence collective.

Configurer la base de données

Puisque le réseau de neurones doit apprendre grâce aux données, nous allons en stocker une représentation dans la base de données. Nous avons juste besoin d'une table pour les nœuds (nœudscaches) et de deux tables pour les connexions (nœudsortie et nœudcache). Créez le fichier *mn.py* et ajoutez-lui une classe nommée *Rechercheur*.

```
# -*- coding: latin-1 -*-
```

```
from math import tanh
from pysqlite2 import dbapi2 as sqlite
```

```
class rechercheur:
```

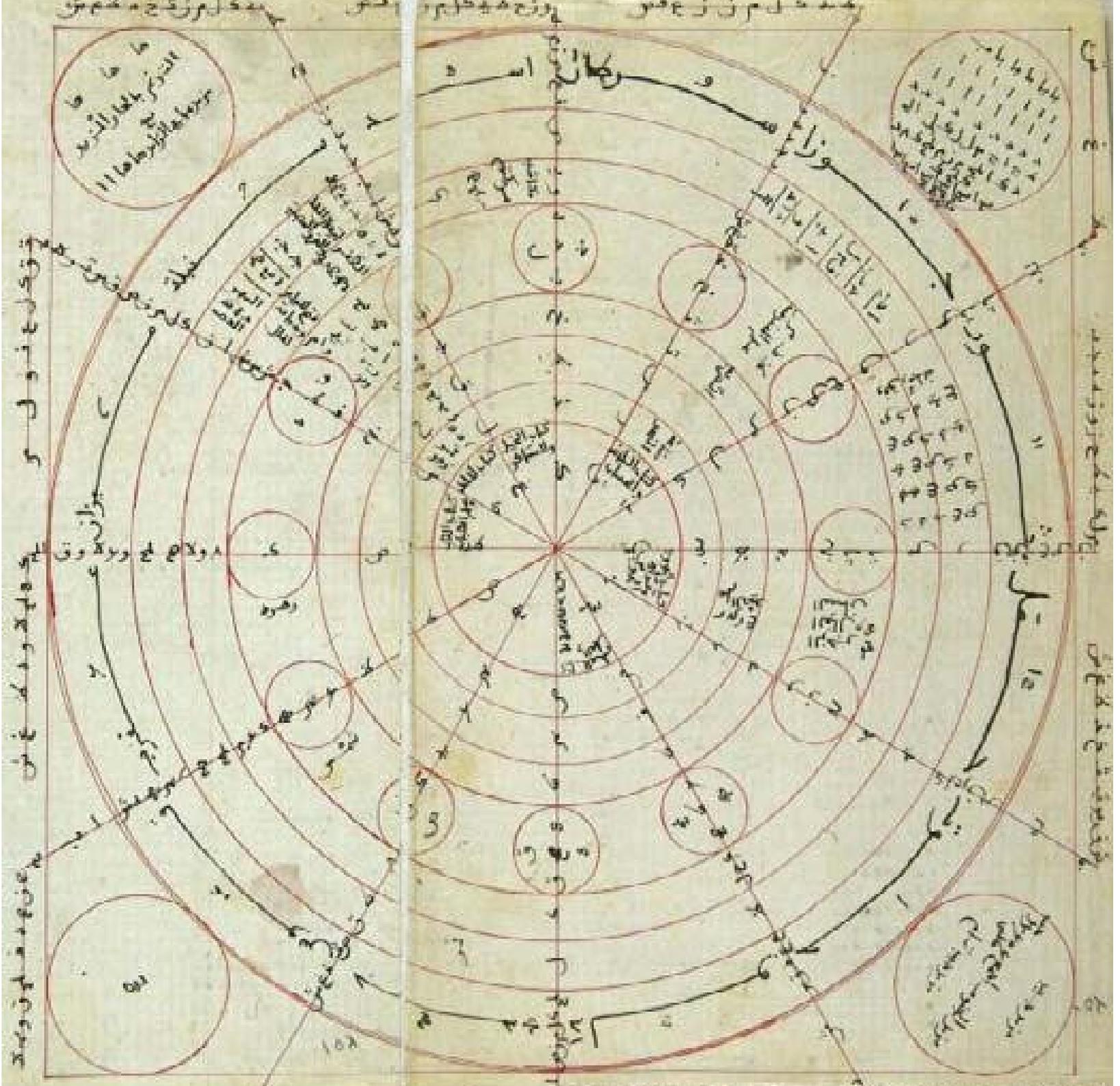
David Link

“Know that the secrets of God and the objects of His science, the subtle realities and the dense realities, the things of above and the things from below, belong to two categories: there are numbers and there are letters.

The secrets of the letters are in the numbers, and the epiphanies of the numbers are in the letters. The numbers are the realities of above, belonging to the spiritual entities. The letters belong to the circle of the material realities and to the becoming.” Aḥmad al-Būn

zā'irjah

- The artefact offered a most astonishing function: **Taking into account the moment in time of the enquiry, it generated a rhymed answer to any question posed.** Possibly Llull was impressed by the fact that even kings placed a high trust in the procedure: “Many distinguished people have shown great interest in using [the zā'irajah] for supernatural information, with the help of the well-known enigmatic operation that goes with it.”
- + 260 → the hand of the human subject



الارض
القمر
الشمس
الزهرة
المريخ
ال木星
الเสาร์

القمر
الشمس
الزهرة
المريخ
ال木星
الเสาร์

عن مركز الارض
عن مركز الشمس
عن مركز القمر

عن مركز الارض
عن مركز الشمس
عن مركز القمر

الارض
القمر
الشمس
الزهرة
المريخ
ال木星
الเสาร์

الارض
القمر
الشمس
الزهرة
المريخ
ال木星
الเสาร์

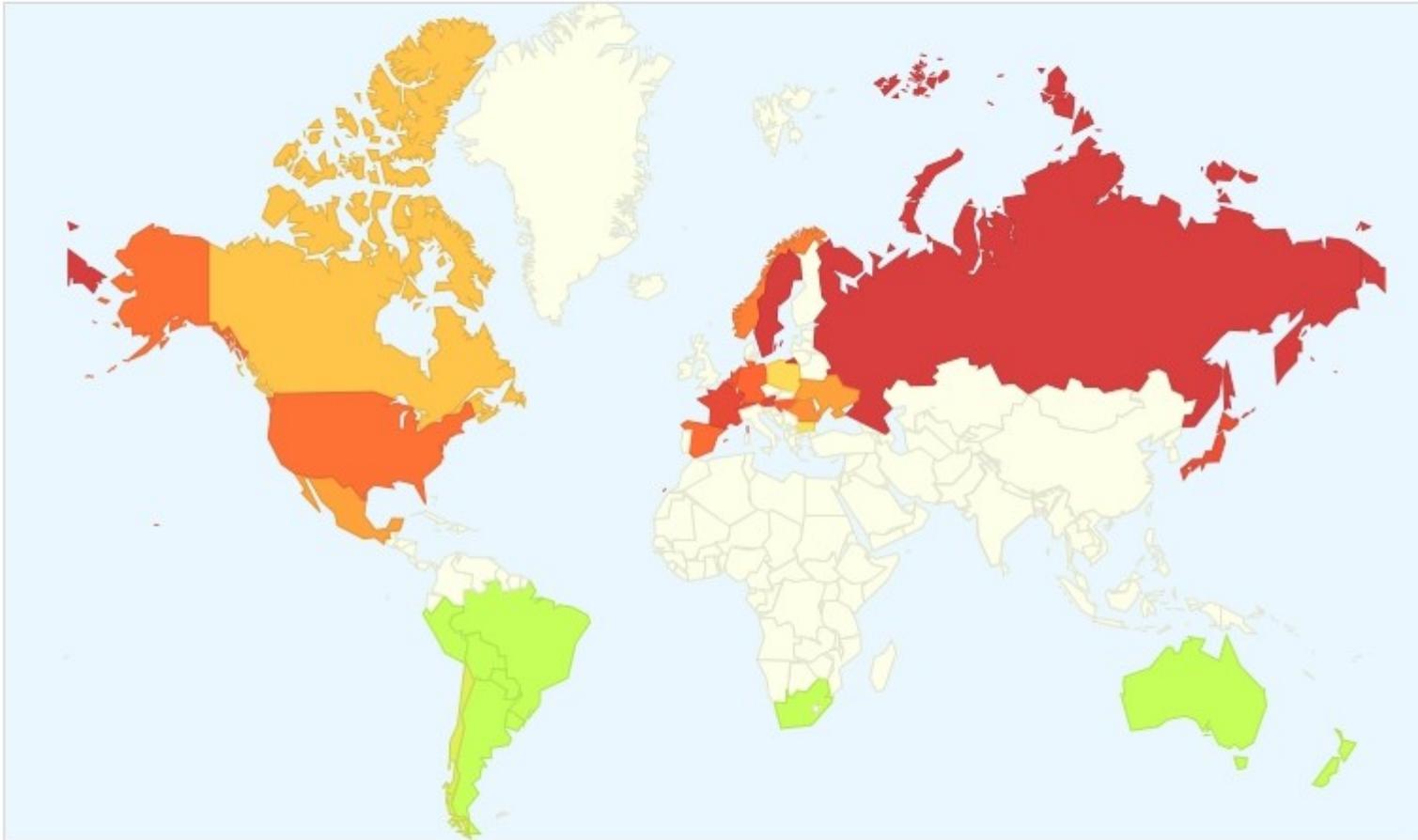


Table on the left side of the page, containing handwritten entries in a grid format. The text is dense and appears to be a list or ledger.

Table on the right side of the page, containing handwritten entries in a grid format. This table is larger and contains more detailed data than the left table.

Explore flu trends around the world

We've found that certain search terms are good indicators of flu activity. Google Flu Trends uses aggregated Google search data to estimate flu activity. [Learn more »](#)



[Download world flu activity data](#) - [Animated flu trends for Google Earth](#) - [Compare flu trends across regions in Public Data Explorer](#)

Autres moteurs

de recherche français Qwant réussit son lancement

Outre les géants Yahoo! et Google, les moteurs de recherche Internet qui sont parvenus à se faire une place aux côtés de Google sont rares. Il y a ceux qui disposent d'une solide assise nationale, comme le russe Yandex, le chinois Baidu et le sud-coréen Naver, dans des pays où Google se casse les dents. Puis ceux qui ont su trouver un créneau original.

Le précédent DuckDuckGo

Aux États-Unis, le moteur de recherche DuckDuckGo attaque ainsi Google sur la question de la vie privée et de la personnalisation de l'information. Il refuse de pister l'internaute pour lui vendre de la publicité ciblée et n'adapte pas les résultats en fonction de ses recherches précédentes. La recette plaît. DuckDuckGo est passé de 15 à

50 millions de requêtes mensuelles en un an.

En France, le grand projet de moteur franco-allemand voulu par Jacques Chirac en 2005 a dérivé en programme de recherche, Quaero. Il a donné naissance à des moteurs spécialisés dans la recherche multimédia. Orange développe « Le Moteur », dont une nouvelle version vient d'être déployée. Une rubrique de recherche d'actualités a été lancée en décembre.

Qwant, de son côté, a choisi de ne pas dupliquer le modèle Google. Les résultats s'affichent en quatre grandes catégories : le Web, le live, le social et le shopping. « Google est le meilleur moteur du monde quand vous savez déjà ce que vous voulez trouver. Qwant est un moteur de découverte, vous y trouvez des choses nouvelles », fait valoir

Éric Léandri, directeur général de la société.

Qwant est en phase de test et promet des améliorations, dans la présentation de ses pages et la taille de son index. Mais déjà, les avis sur la pertinence du moteur de recherche sont en majorité positifs, tant sur les sites spécialisés que sur les réseaux sociaux.

Pour parvenir à ce résultat, Qwant a dû se montrer ingénieux. Il tient compte des liens publiés sur réseaux sociaux, dont Twitter, pour trouver rapidement des liens pertinents et alimenter son index. La société à l'origine du projet, Pertimm, fournit déjà des moteurs de recherche professionnels utilisés par Auchan, Meetic et PagesJaunes. Le développement supplémentaire a nécessité « quelques millions d'euros ». ■

« Google est le meilleur du monde quand vous savez déjà ce que vous voulez trouver »

ÉRIC LÉANDRI, DIRECTEUR GÉNÉRAL DE QWANT

Définition problématique

- Wikipedia

de $\mathcal{P}(E)$ fait correspondre son complémentaire dans E est une algèbre de Boole \blacklozenge .
 Les propositions logiques ont une structure analogue : « et » joue le rôle de \top ,
 « ou » celui de \perp , l'application $x \mapsto \bar{x}$ est ici la négation (le signe $-$, quand
 il s'agit de propositions logiques, est remplacé par le signe \leftrightarrow).

\blacklozenge Réciproquement, pour toute algèbre de Boole, il existe un ensemble F tel que E soit isomorphe à une partie de $\mathcal{P}(F)$.

Anneau de Boole : un ensemble E , qui a structure d'anneau pour les deux lois de composition interne $+$ et \times , est un anneau de Boole si, quel que soit l'élément x de E , $x \times x = x$ (tout élément x de E est idempotent pour la loi \times).
 L'ensemble $\mathcal{P}(E)$ des parties d'un ensemble E , muni des deux lois de composition interne, différence symétrique et intersection, est un anneau de Boole.

Treillis de Boole : voir Treillis.

ALGÈBRE pages 373 et 374.

Équation algébrique : voir Équations.

Mesure algébrique

Soient A et B deux points situés sur un axe Ox . On appelle mesure algébrique du segment AB et on note \overline{AB} le nombre réel (positif ou négatif) égal à la différence entre l'abscisse de B et celle de A : $\overline{AB} = x_B - x_A$.

Exemples

$$\begin{array}{l} \text{Abscisse de } B = 3 \\ \text{Abscisse de } A = 1 \end{array} \left. \vphantom{\begin{array}{l} \text{Abscisse de } B = 3 \\ \text{Abscisse de } A = 1 \end{array}} \right\} \overline{AB} = 3 - 1 = 2$$

$$\begin{array}{l} \text{Abscisse de } A = 5 \\ \text{Abscisse de } B = 2 \end{array} \left. \vphantom{\begin{array}{l} \text{Abscisse de } A = 5 \\ \text{Abscisse de } B = 2 \end{array}} \right\} \overline{AB} = 2 - 5 = -3$$

La mesure algébrique de AB et celle de BA sont opposées : $\overline{AB} = -\overline{BA}$.
 Si A, B, C sont trois points d'un même axe, les mesures algébriques de AB , de BC et de CA sont liées par la relation :

$$\overline{AB} + \overline{BC} + \overline{CA} = 0.$$

Cette relation est appelée relation de Chasles.

Nombre algébrique : voir l'article « les Nombres », page 373.

ALGORITHME pages 179 et 180.

En arithmétique, en informatique, etc. : suite d'opérations permettant de résoudre un problème particulier.

Exemple : algorithme d'Euclide permettant de trouver le plus grand commun diviseur (p.g.c.d.) de deux nombres a et b . On commence par diviser a par b ,

Figure 4

$$\begin{array}{r|l}
 6\ 52\ 32\ 45 & 25 \\
 2\ 52 & 45 \times 5 = 225 \\
 \hline
 2\ 25 & 505 \times 5 = 2525 \\
 27\ 32 &
 \end{array}$$

Figure 5

$$\begin{array}{r|l}
 6\ 52\ 32\ 45 & 2554 \\
 2\ 52 & 45 \times 5 = 225 \\
 \hline
 2\ 25 & 505 \times 5 = 2525 \\
 27\ 32 & 5104 \times 4 = 20416 \\
 \hline
 25\ 25 & \\
 2\ 07\ 45 &
 \end{array}$$

puis b par le reste de cette première opération, puis ce dernier reste par le second, etc. On arrive, après un certain nombre d'opérations, à un reste nul. Le reste précédent est le p.g.c.d.

Calculs faits pour trouver le p.g.c.d. de 75 et de 45 :

$$75 : 45 = 1, \text{ reste } 30 ;$$

$$45 : 30 = 1, \text{ reste } 15 ;$$

$$30 : 15 = 2, \text{ reste } 0.$$

Le p.g.c.d. de 75 et de 45 est donc 15.

Algorithme de l'extraction de la racine carrée

Supposons vouloir extraire la racine carrée du nombre 6 523 245.

On dispose une opération analogue à une division.

On sépare le nombre en tranches de deux chiffres à partir de la droite (fig. 1).

On cherche le plus grand nombre dont le carré soit inférieur à la tranche de gauche (ici : 6) ; ce nombre est 2.

On inscrit ce nombre à droite et on soustrait son carré de la tranche de gauche (ici : $6 - 4 = 2$) (fig. 2).

On place à côté du résultat obtenu la tranche suivante de deux chiffres (52).

On multiplie le nombre situé au-dessus du trait horizontal par 2 et on l'inscrit en dessous du trait horizontal (ici : 4).

On cherche quel est le plus grand chiffre x que l'on doit inscrire à la suite de 4 pour que $4x \times x$ soit plus petit que 252 ; on trouve $45 \times 5 = 225$, qui est bien inférieur à 252 (alors que $46 \times 6 = 276$ est supérieur à 252) (fig. 3).

On inscrit 5 à côté du 2 (fig. 4). On soustrait 225 de 252 ; on trouve 27. On écrit à côté de 27 la tranche suivante de deux chiffres : 32. On multiplie 25 par 2 : 50 et on cherche le plus grand chiffre x que l'on doit inscrire à la suite de 50 pour que $50x \times x$ soit inférieur à 2 732 ; c'est 5.

On l'écrit à la suite de 25 : 255 (fig. 5).

On retranche 2 525 de 2 732 et on obtient 207.

On inscrit à côté de 207 la tranche suivante de deux chiffres : 45. On double $255 = 510$ et on cherche quel est le chiffre x qu'il faut mettre à la suite de 510 pour que $510x \times x$ soit inférieur à 20 745 ; c'est 4. On l'inscrit après 255 : 2 554.

Il n'y a plus de tranche de deux chiffres à utiliser ; on s'arrête donc.

La racine carrée à une unité près de 6 523 245 est 2 554.

Évolution des mathématiques

- Rohrer, New Mathematics and the subject of the variable

p 326 → Particularly the polemics → p 326 is met by the world.

P 328 → Similar to the reform → p 329 indispensable means of expression

Évolution des mathématiques

- Jacques Roubaud, Mathématiques

p9 1. Il y avait trois issues → p11 ils ne s'étaient pas attendus à cela.

p21 5 Ce qui provoquait la stupeur inquiète →
p23 je me trouvais là au début, parmi eux, moi aussi.

Algol

ALGOL (short for ALGORithmic Language) is a family of computer programming languages originally developed in the mid-1950s which greatly influenced many other languages and was the **standard method for algorithm description** ...

src: <http://en.wikipedia.org/wiki/ALGOL>

Algol et Oulipo

François Le Lionnais, Ivresse Algolique

Intro p213 → 218

Poème p219 → 220

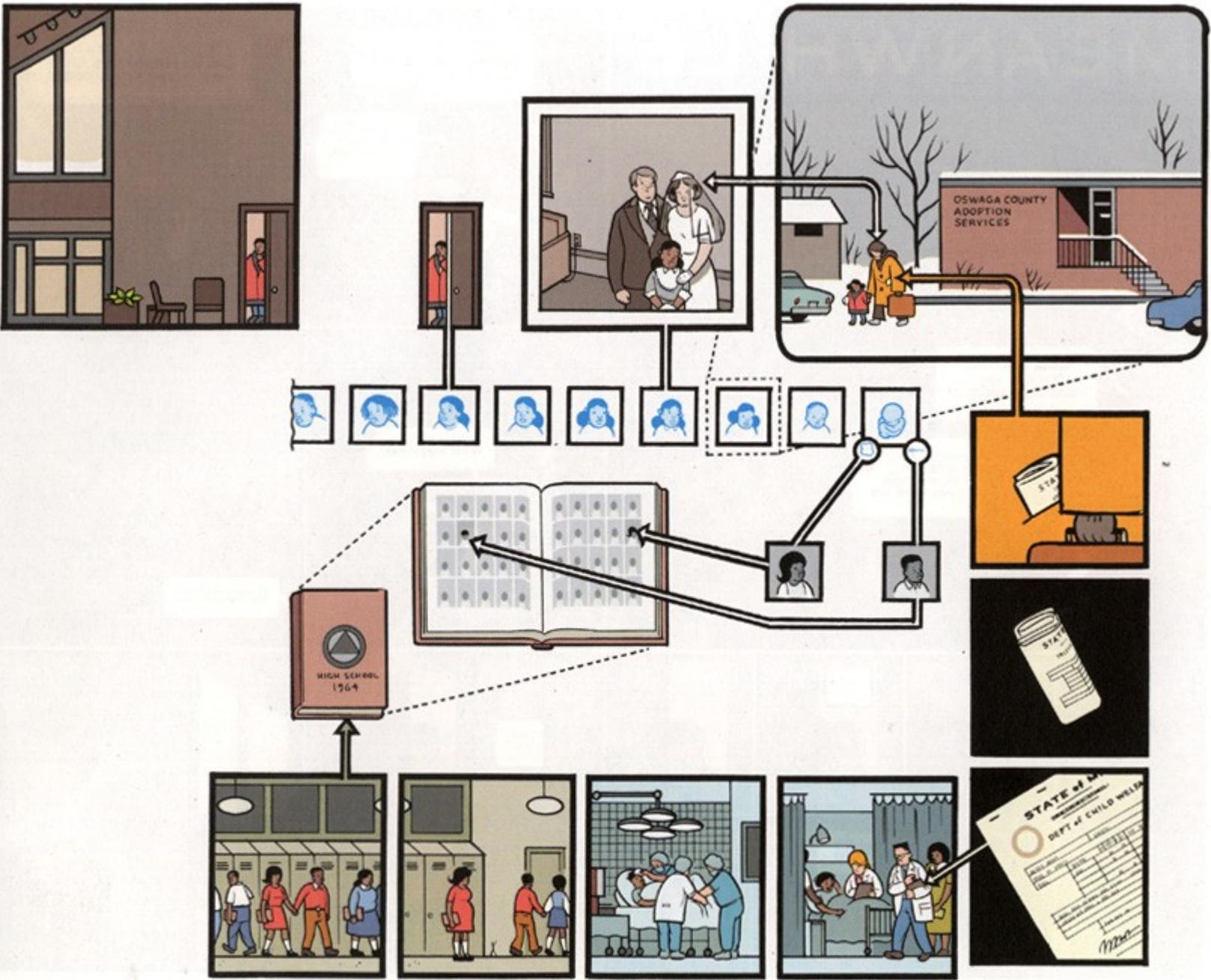
- Michael Murtaugh:

Pask: “twoness” is a notion that can be abstracted from two objects.

- But counting is also always performing a classification (sélectionner deux poires dans un ensemble de fruits)

Nouveau partage entre la lettre et le chiffre.

- L'algorithme n'est plus seulement une méthode de calcul, mais un script.
- La narration et le calcul se combinent dans le code.
- Importance du jeu comme modèle.



La transformée de Fourier



La transformée de Fourier

La TF est un processus mathématique permettant de décomposer un signal complexe, fonction du temps, mais pas forcément périodique en une somme de signaux simples de fréquences connus donc périodiques.

La transformée de Fourier

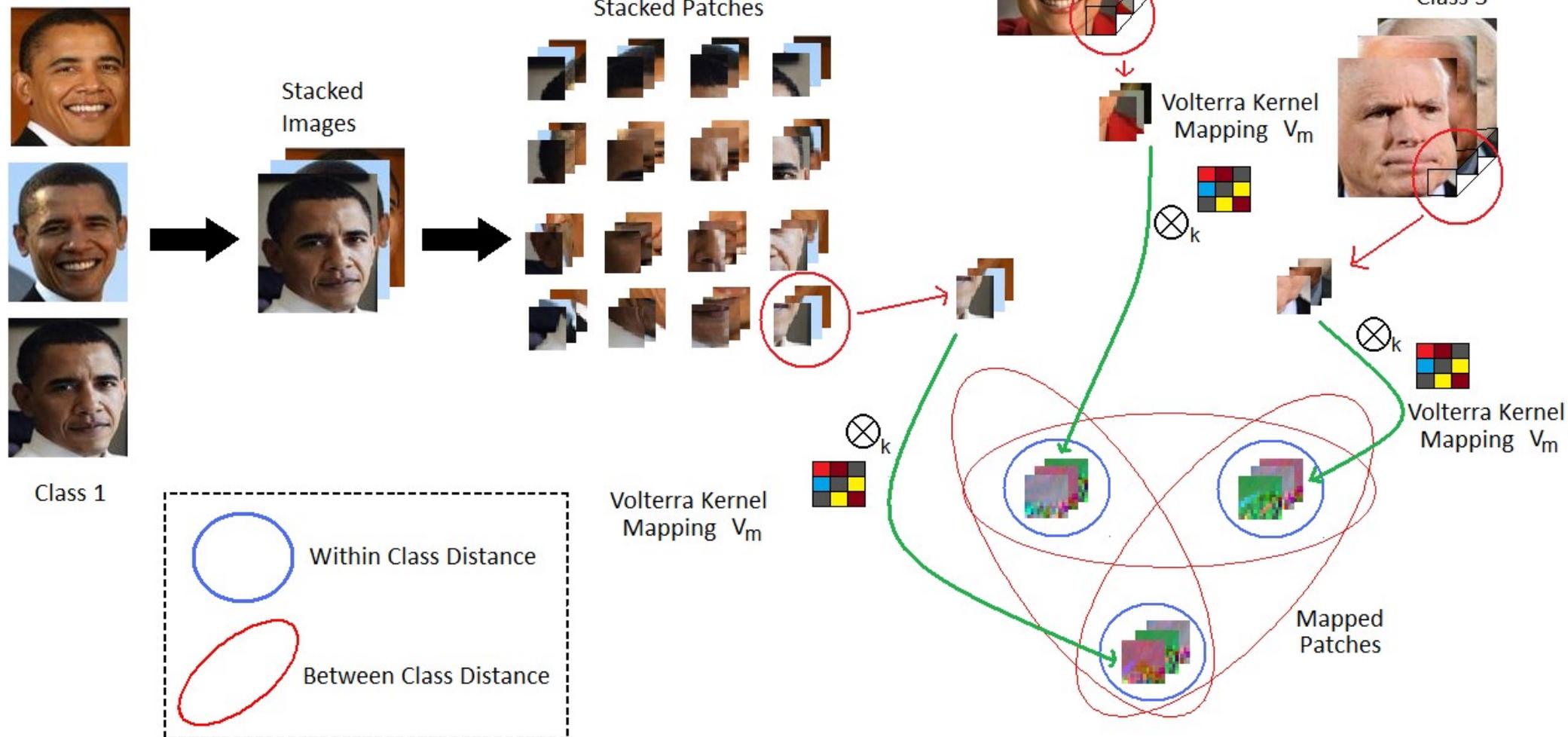
Tukey arrived at the basic reduction while in a meeting of President Kennedy's Science Advisory Committee where among the topics of discussions were techniques for off-shore detection of nuclear tests in the Soviet Union. Ratification of a proposed United States/Soviet Union nuclear test ban depended upon the development of a method for detecting the tests without actually visiting the Soviet nuclear facilities.

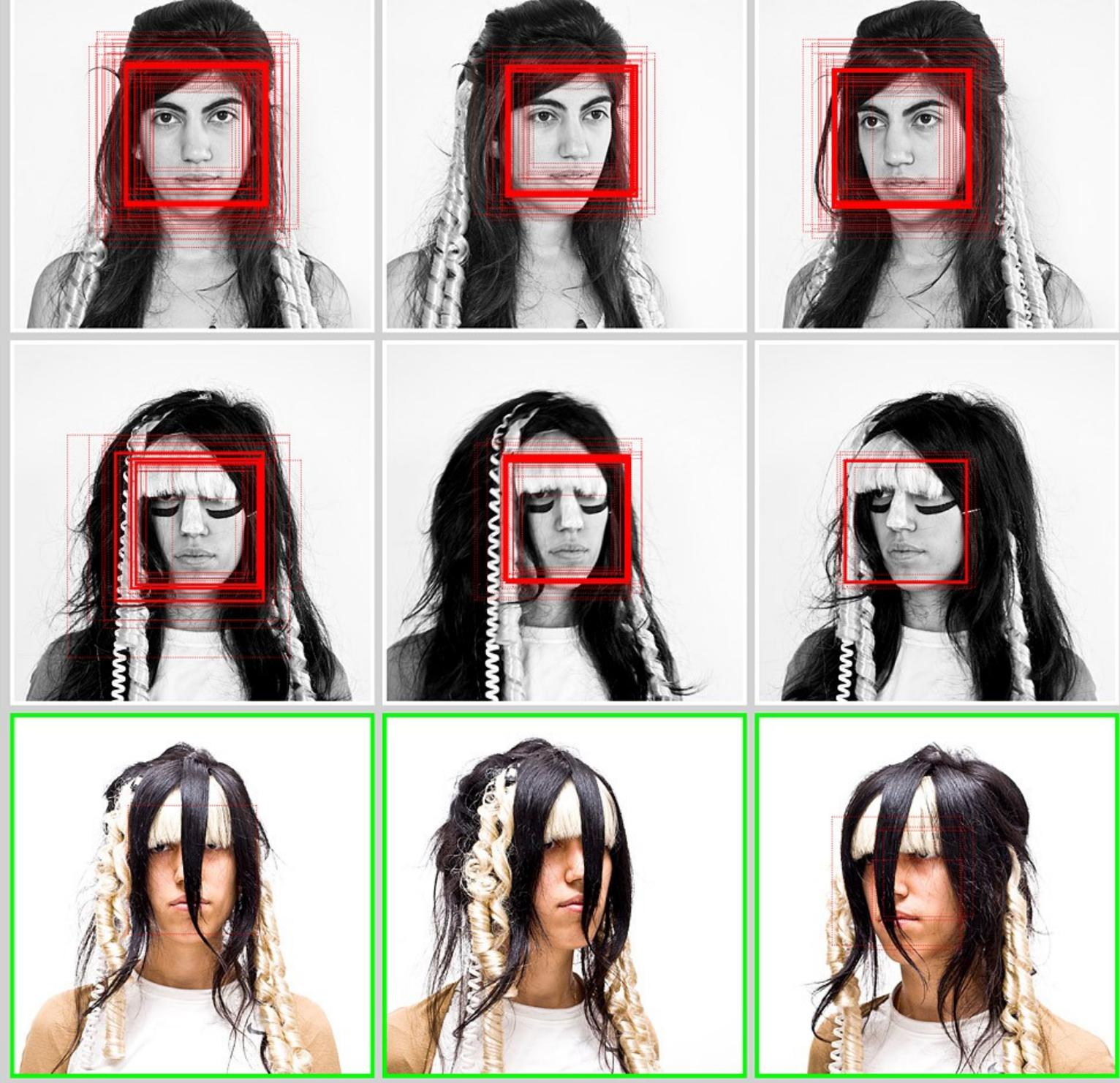
Face detection



<http://vimeo.com/12774628>

VOLTERRA KERNEL TRAINING





Propriété intellectuelle

I am told that the courts are trying to make a distinction between mathematical algorithms and nonmathematical algorithms.

To a computer scientist, this makes no sense, because every algorithm is as mathematical as anything could be.

Donald Knuth

Propriété intellectuelle

- Brevets et algorithmes
- Différence entre copyright et brevet
- Problèmes de définition

- Sources:
- http://en.wikipedia.org/wiki/List_of_software_patents
- <http://programmers.stackexchange.com/questions/32482/can-an-algorithm-be-patented>

Patent Absurdity

- <http://patentabsurdity.com/watch.html>