Diviser pour régner

Divide et impera attribuée à Philippe II de Macédoine IV$^{\text{ième}}$ siècle avant J.C

Jean-Marc.Vincent@univ-grenoble-alpes.fr[1]

[1] Laboratoire LIG
Équipe-Projet INRIA POLARIS
Université Grenoble-Alpes

Algorithmique et Modélisation
L3-INFO

DIVISER POUR RÉGNER

UN MODE DE PENSÉE

La solution d'un problème de $\mathscr{P}_n$ sur une donnée de taille n s'exprime comme la composition de solutions de sous-problèmes de même nature de taille plus petite. Si la taille est inférieure à une borne on applique une méthode directe (en général moins coûteuse).

Un mode de pensée

La solution d'un problème de $\mathscr{P}_n$ sur une donnée de taille n s'exprime comme la composition de solutions de sous-problèmes de même nature de taille plus petite. Si la taille est inférieure à une borne on applique une méthode directe (en général moins coûteuse).

Schéma basé sur l'induction

Preuve de la correction facile

- correction partielle
  cas de base, résolution des sous-problèmes $\Rightarrow$ solution du problème
- terminaison (variant)

Un mode de pensée

La solution d'un problème de $\mathscr{P}_n$ sur une donnée de taille n s'exprime comme la composition de solutions de sous-problèmes de même nature de taille plus petite. Si la taille est inférieure à une borne on applique une méthode directe (en général moins coûteuse).

Schéma basé sur l'induction

Preuve de la correction facile

- correction partielle
  cas de base, résolution des sous-problèmes $\Rightarrow$ solution du problème
- terminaison (variant)

Mise en œvre simple (opérationnel)

langage admettant la récursivité
dérécursification possible

Un mode de pensée

La solution d'un problème de $\mathscr{P}_n$ sur une donnée de taille n s'exprime comme la composition de solutions de sous-problèmes de même nature de taille plus petite. Si la taille est inférieure à une borne on applique une méthode directe (en général moins coûteuse).

Schéma basé sur l'induction

Preuve de la correction facile

- correction partielle
  cas de base, résolution des sous-problèmes $\Rightarrow$ solution du problème
- terminaison (variant)

Mise en œvre simple (opérationnel)

langage admettant la récursivité
dérécursification possible

Complexité

Le schéma correspond au Master-Theorem

$$\begin{cases} C(1) = 1; \\ C(n) = aC(\frac{n}{b}) + c(n), \text{ pour } n \geqslant 1, \end{cases}$$

$$C(n) = \begin{cases} \Theta(n^{\log_b a}) & \text{si } c(n) = O(n^{\log_b a-\varepsilon}) \\ \Theta(n^{\log_b a}\log n) & \text{si } c(n) = \Theta(n^{\log_b a}) \\ \Theta(c(n)) & \text{si } c(n) = \Omega(n^{\log_b a+\varepsilon}) \end{cases}$$

EXEMPLE LIMITE

Lorsque l'on a un seul sous problème de taille plus petite (avec un facteur de division)

Recherche dichotomique

```
Recherche_dichotomique (T,x,i,j)
    Données: T tableau trié, x élément de T, i ⩽ j, indices de T, T[i] ⩽ x ⩽ T[j]
    Résultat: L'indice de x dans T
    m = (i+j)/2
    if T[milieu] = x
      └ Retourne milieu
    else if T[milieu] > x
      └ Retourne Recherche_dichotomique (T, x, i, milieu)
    else
      └ Retourne Recherche_dichotomique (T, x, milieu, j)
```

EXEMPLE LIMITE

Lorsque l'on a un seul sous problème de taille plus petite (avec un facteur de division)

Recherche dichotomique

`Recherche_dichotomique` (T,x,i,j)

Données: T tableau **trié**, x élément de T, $i \leqslant j$, indices de T, $T[i] \leqslant x \leqslant T[j]$

Résultat: L'indice de x dans T

$m = \frac{i+j}{2}$

if $T[milieu] = x$

  `Retourne` *milieu*

else if $T[milieu] > x$

  `Retourne Recherche_dichotomique` $(T, x, i, milieu)$

else

  `Retourne Recherche_dichotomique` $(T, x, milieu, j)$

$$C(n) = 1 \times C(\frac{n}{2}) + 1 \text{ d'où } C(n) = \Theta(\log n)$$

réduction logarithmique de la complexité (par rapport à une recherche séquentielle en $\mathcal{O}(n)$)

- Recherche de rupture dans un tableau circulairement trié
- Recherche de pic (tableau avec un seul pic)
- Exponentiation rapide ...

DIVISER POUR RÉGNER

ARCHÉTYPE DE DIVISER POUR RÉGNER (1) : LE "MERGE SORT"

Tri partition/fusion : division en 2 sous-problèmes de taille $\frac{n}{2}$

```
Partition_Fusion (T)
   Données: T tableau d'éléments comparables
   Résultat: Les éléments de T rangés par ordre croissant
   if longueur(T) ⩽ 1
      Retourne T
   else
      T1 = Partition_Fusion(Moitie_Gauche(T))
      T2 = Partition_Fusion(Moitie_Droite(T))
      Retourne Fusion (T1, T2)
```

ARCHÉTYPE DE DIVISER POUR RÉGNER (1) : LE "MERGE SORT"

Tri partition/fusion : division en 2 sous-problèmes de taille $\frac{n}{2}$

```
Partition_Fusion (T)
    Données: T tableau d'éléments comparables
    Résultat: Les éléments de T rangés par ordre croissant

    if longueur(T) ⩽ 1
     └ Retourne T
    else
     | T1 = Partition_Fusion(Moitie_Gauche(T))
     | T2 = Partition_Fusion(Moitie_Droite(T))
     └ Retourne Fusion (T1, T2)
```

$$C(n) = 2C(\frac{n}{2}) + \Theta(n) \text{ d'où } C(n) = \Theta(n \log n)$$

réduction de la complexité (par rapport à tri "élémentaire" en $\mathcal{O}(n^2)$)

ARCHÉTYPE DE DIVISER POUR RÉGNER (2) : LE "QUICK SORT"

Tri par segmentation (tri rapide) : division en 2 sous-problèmes de taille dépendant des données $n_1 + n_2 = n - 1$

```
Tri_par_Segmentation (T)
   Données: T tableau d'éléments comparables
   Résultat: Les éléments de T rangés par ordre croissant
   if longueur(T) ⩽ 1
     Retourne T
   else
     x = Choisir(T)
     Retourne Assemble(Plus_Petits(T, x), x, Plus_Grands(T, x))
```

ARCHÉTYPE DE DIVISER POUR RÉGNER (2) : LE "QUICK SORT"

Tri par segmentation (tri rapide) : division en 2 sous-problèmes de taille dépendant des données $n_1 + n_2 = n - 1$

```
Tri_par_Segmentation (T)
   Données: T tableau d'éléments comparables
   Résultat: Les éléments de T rangés par ordre croissant
   if longueur(T) ⩽ 1
      Retourne T
   else
      x = Choisir(T)
      Retourne Assemble(Plus_Petits(T, x), x, Plus_Grands(T, x))
```

Coût **moyen**

$$C_m(n) = \frac{1}{n}\sum_{i=0}^{n-1}(C_m(i) + C_m(n-1-i)) + \Theta(n) \text{ d'où } C_m(n) = \Theta(n \log n)$$

réduction de la complexité (par rapport à tri "élémentaire" en $\mathcal{O}(n^2)$)

DIVISER POUR RÉGNER

Un peu de calcul

Multiplication d'entiers

$$132 \times 21 = 2772$$

Multiplication posée

$$\begin{array}{r} 1\ 3\ 2 \\ \times \quad 2\ 1 \\ \hline 1\ 3\ 2 \\ 2\ 6\ 4\ \cdot \\ \hline 2\ 7\ 7\ 2 \end{array}$$

Un peu de calcul

Multiplication d'entiers

$$132 \times 21 = 2772$$

Multiplication posée

$$\begin{array}{r} 1\,3\,2 \\ \times \quad 2\,1 \\ \hline 1\,3\,2 \\ 2\,6\,4\,\cdot \\ \hline 2\,7\,7\,2 \end{array}$$

$$\begin{aligned} (2.10^0 + 3.10^1 + 1.10^2) \times (1.10^0 + 2.10^1) &= (2 \times 1).10^0 \\ &\quad +(2 \times 2 + 3 \times 1).10^1 \\ &\quad +(1 \times 1 + 3 \times 2).10^2 \\ &\quad +(1 \times 2 + 3 \times 0).10^3 \\ &= 2.10^0 + 7.10^1 + 7.10^2 + 2.10^3 \\ &= 2772 \end{aligned}$$

Produit en "croix"

Un peu de calcul

Multiplication d'entiers

$$132 \times 21 = 2772$$

$$132 \times 81 = 10692$$

Multiplication posée

$$\begin{array}{rrrr} & 1 & 3 & 2 \\ \times & & 2 & 1 \\ \hline & 1 & 3 & 2 \\ 2 & 6 & 4 & \cdot \\ \hline 2 & 7 & 7 & 2 \end{array} \qquad \begin{array}{rrrrr} & & 1 & 3 & 2 \\ & \times & & 8 & 1 \\ \hline & & 1 & 3 & 2 \\ 1 & 0 & 5 & 6 & \cdot \\ \hline 1 & 0 & 6 & 9 & 2 \end{array}$$

$$\begin{aligned} (2.10^0 + 3.10^1 + 1.10^2) \times (1.10^0 + 2.10^1) &= (2 \times 1).10^0 \\ &\quad +(2 \times 2 + 3 \times 1).10^1 \\ &\quad +(1 \times 1 + 3 \times 2).10^2 \\ &\quad +(1 \times 2 + 3 \times 0).10^3 \\ &= 2.10^0 + 7.10^1 + 7.10^2 + 2.10^3 \\ &= 2772 \end{aligned}$$

Produit en "croix"

Un peu de calcul

Multiplication d'entiers

$$132 \times 21 = 2772$$

$$132 \times 81 = 10692$$

Multiplication posée

$$\begin{array}{rrrr} & 1 & 3 & 2 \\ \times & & 2 & 1 \\ \hline & 1 & 3 & 2 \\ 2 & 6 & 4 & \cdot \\ \hline 2 & 7 & 7 & 2 \end{array} \qquad \begin{array}{rrrrr} & & 1 & 3 & 2 \\ & \times & & 8 & 1 \\ \hline & & 1 & 3 & 2 \\ 1 & 0 & 5 & 6 & \cdot \\ \hline 1 & 0 & 6 & 9 & 2 \end{array}$$

$$\begin{aligned} (2.10^0 + 3.10^1 + 1.10^2) \times (1.10^0 + 2.10^1) &= (2 \times 1).10^0 \\ &\quad +(2 \times 8 + 3 \times 1).10^1 \\ &\quad +(1 \times 1 + 3 \times 8).10^2 \\ &\quad +(1 \times 8 + 3 \times 0).10^3 \\ &= 2.10^0 + 19.10^1 + 25.10^2 + 8.10^3 \\ &= 10692 \end{aligned}$$

Produit en "croix"
Propagation de retenue

UN PEU DE CALCUL (SUITE)

Représentation d'un nombre entier : base β

$$A = a_0\beta^0 + a_1\beta^1 + \cdots a_{n-1}\beta^{n-1} = \sum_{i=0}^{n-1} a_i\beta^i \text{ avec } 0 \leqslant a_i < \beta$$

$$B = b_0\beta^0 + b_1\beta^1 + \cdots b_{n-1}\beta^{n-1} = \sum_{j=0}^{n-1} b_j\beta^j \text{ avec } 0 \leqslant b_j < \beta$$

Un peu de calcul (suite)

Représentation d'un nombre entier : base β

$$A = a_0\beta^0 + a_1\beta^1 + \cdots a_{n-1}\beta^{n-1} = \sum_{i=0}^{n-1} a_i\beta^i \text{ avec } 0 \leqslant a_i < \beta$$

$$B = b_0\beta^0 + b_1\beta^1 + \cdots b_{n-1}\beta^{n-1} = \sum_{j=0}^{n-1} b_j\beta^j \text{ avec } 0 \leqslant b_j < \beta$$

$$A \times B = \left(\sum_{i=0}^{n-1} a_i\beta^i\right) . \left(\sum_{j=0}^{n-1} b_j\beta^j\right) = \sum_{k=0}^{??} c_k\beta^k \quad c_k = \sum_{i+j=k} a_i.b_j$$

Borne de la somme : $??= (n-1) + (n-1) = 2n-2$

Attention : les retenues ne sont pas encore prises en compte

Un peu de calcul (suite)

Représentation d'un nombre entier : base β

$$
\begin{aligned}
A &= a_0\beta^0 + a_1\beta^1 + \cdots a_{n-1}\beta^{n-1} = \sum_{i=0}^{n-1} a_i\beta^i \text{ avec } 0 \leqslant a_i < \beta \\
B &= b_0\beta^0 + b_1\beta^1 + \cdots b_{n-1}\beta^{n-1} = \sum_{j=0}^{n-1} b_j\beta^j \text{ avec } 0 \leqslant b_j < \beta
\end{aligned}
$$

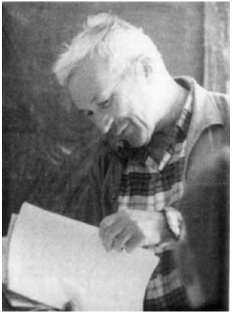
$$
A \times B = \left(\sum_{i=0}^{n-1} a_i\beta^i\right) . \left(\sum_{j=0}^{n-1} b_j\beta^j\right) = \sum_{k=0}^{??} c_k\beta^k \quad c_k = \sum_{i+j=k} a_i.b_j
$$

Borne de la somme : $?? = (n-1) + (n-1) = 2n-2$

Attention : les retenues ne sont pas encore prises en compte

Conjecture : la complexité de la multiplication est en $\mathcal{O}(n^2)$ (Kolmogorov $\sim$ 1956)

ANDREÏ KOLMOGOROV (1903-1987)



Mathématicien russe dont le nom est principalement attaché à l'axiomatisation du calcul des probabilités dans le cadre de la théorie des ensembles. Fils d'un agronome, Andreï Nikolaïevitch Kolmogorov est né à Tambov. Il entra à dix-sept ans à l'université de Moscou, où il suivit des cours de N. Lusin et de P. Urysohn ; chercheur associé à cette université depuis 1925, il y devint professeur en 1931 et directeur du département de mathématiques deux ans plus tard. En 1939, il fut élu à l'Académie des sciences de l'U.R.S.S.

Les premiers travaux de Kolmogorov portent sur les fonctions de variable réelle (séries trigonométriques, opérations sur les ensembles) ; en 1922, il a construit un exemple célèbre de fonction intégrable dont la série de Fourier est divergente en tout point, ce qui relançait le problème de la convergence des séries de Fourier. Quelques années plus tard, il étendit la sphère de ses recherches à la logique mathématique et aux problèmes de fondements. À partir de 1925, en collaboration avec A. Khintchine, Kolmogorov a étudié les problèmes de convergence de séries d'éléments aléatoires, sur lesquels il a publié de nombreux articles devenus classiques. Son mémoire fondamental, Théorie générale de la mesure et théorie des probabilités (1929), donne la première construction axiomatique du calcul des probabilités fondée sur la théorie de la mesure ; il développa ses idées dans l'ouvrage Grundbegriffe der Wahrscheinlichkeitsrechnung (trad. angl. Foundations of the Theory of Probability, 1950), publié en 1933. Avec son ouvrage Méthodes analytiques de théorie des probabilités, Kolmogorov est un des fondateurs de la théorie des processus stochastiques. Il a étudié plus spécialement ceux qui sont connus de nos jours sous le nom de processus de Markov où deux systèmes d'équations aux dérivées partielles portent son nom ; cette théorie a d'importantes applications en physique (mouvement brownien, diffusion). Mentionnons aussi des recherches très importantes sur les processus aléatoires stationnaires, dont Wiener a souligné le rôle dans la théorie statistique de l'information sur laquelle s'appuie, pour une part, la cybernétique. Kolmogorov a également fait des recherches en topologie, géométrie, analyse fonctionnelle et approximation optimale des fonctions. Depuis 1950, il a publié des travaux sur la théorie de l'information, la mécanique classique et la théorie des automates finis. Il a consacré ses dernières années à des problèmes d'enseignement des mathématiques et a publié plusieurs ouvrages de pédagogie à l'usage des parents et des enseignants. Il termina sa vie à Moscou.

UN PEU DE CALCUL (SUITE ET FIN)

Cas $n = 2$ (nombre avec 2 chiffres)

$$(a_0 + a_1.\beta)(b_0 + b_1.\beta) \quad = \quad a_0 b_0 + (a_0 b_1 + a_1 b_0).\beta + a_1 b_1 \beta^2$$

UN PEU DE CALCUL (SUITE ET FIN)

Cas $n = 2$ (nombre avec 2 chiffres)

$$\begin{aligned}(a_0 + a_1.\beta)(b_0 + b_1.\beta) &= a_0 b_0 + (a_0 b_1 + a_1 b_0).\beta + a_1 b_1 \beta^2 \\ &= a_0 b_0 + (a_0 b_0 + a_1 b_1 - (a_1 - a_0)(b_1 - b_0)).\beta + a_1 b_1 \beta^2\end{aligned}$$

Nombre de multiplications : **4→3**

UN PEU DE CALCUL (SUITE ET FIN)

Cas $n = 2$ (nombre avec 2 chiffres)

$$\begin{aligned}(a_0 + a_1.\beta)(b_0 + b_1.\beta) &= a_0b_0 + (a_0b_1 + a_1b_0).\beta + a_1b_1\beta^2 \\ &= a_0b_0 + (a_0b_0 + a_1b_1 - (a_1 - a_0)(b_1 - b_0)).\beta + a_1b_1\beta^2\end{aligned}$$

Nombre de multiplications : **4→3** Généralisation n pair (puissance de 2)

$$\begin{aligned}(A_0 + A_1.\beta^{\frac{n}{2}})(B_0 + B_1.\beta^{\frac{n}{2}}) &= A_0B_0 \\ &\quad + (A_0B_0 + A_1B_1 - (A_1 - A_0)(B_1 - B_0)).\beta^{\frac{n}{2}} \\ &\quad + A_1B_1\beta^n\end{aligned}$$

$$C(n) = 3C(\frac{n}{2}) + \Theta(n) \text{ d'où } C(n) = \Theta(n^{\log_2 3}) \simeq \Theta(n^{1,58})$$

Le traitement des retenues va être en $\Theta(n)$

Un peu de calcul (suite et fin)

Cas $n = 2$ (nombre avec 2 chiffres)

$$\begin{aligned}(a_0 + a_1.\beta)(b_0 + b_1.\beta) &= a_0 b_0 + (a_0 b_1 + a_1 b_0).\beta + a_1 b_1 \beta^2 \\ &= a_0 b_0 + (a_0 b_0 + a_1 b_1 - (a_1 - a_0)(b_1 - b_0)).\beta + a_1 b_1 \beta^2\end{aligned}$$

Nombre de multiplications : **4→3** Généralisation n pair (puissance de 2)

$$\begin{aligned}(A_0 + A_1.\beta^{\frac{n}{2}})(B_0 + B_1.\beta^{\frac{n}{2}}) = \; & A_0 B_0 \\ & + (A_0 B_0 + A_1 B_1 - (A_1 - A_0)(B_1 - B_0)).\beta^{\frac{n}{2}} \\ & + A_1 B_1 \beta^n\end{aligned}$$

$$C(n) = 3C(\frac{n}{2}) + \Theta(n) \text{ d'où } C(n) = \Theta(n^{\log_2 3}) \simeq \Theta(n^{1,58})$$

Le traitement des retenues va être en $\Theta(n)$

Algorithme de Karatsuba (1960)

Multiplication de matrices

Multiplication de matrices

$$A = ((a_{i,j})) \quad B = ((b_{i,j})) \quad \text{produit } C = ((c_{i,j})) \text{ avec } c_{i,j} = \sum_k a_{i,k} b_{k,j}$$

Complexité de l'algorithme $\Theta(n^3)$

MULTIPLICATION DE MATRICES (DIVISER POUR RÉGNER)

Multiplication de matrices : Algorithme de Strassen (1969) (biographie)

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} = \begin{pmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{pmatrix}$$

où

$$\begin{aligned}
M_1 &= (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2}), \\
M_2 &= (A_{2,1} + A_{2,2})B_{1,1}, \\
M_3 &= A_{1,1}(B_{1,2} - B_{2,2}), \\
M_4 &= A_{2,2}(B_{2,1} - B_{1,1}), \\
M_5 &= (A_{1,1} + A_{1,2})B_{2,2}, \\
M_6 &= (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2}), \\
M_7 &= (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2}).
\end{aligned}$$

$$C(n) = 7C(\frac{n}{2}) + \Theta(n) \text{ d'où } C(n) = \Theta(n^{\log_2 7}) = \Theta(n^{2.8})$$

réduction de la complexité par rapport à la multiplication "élémentaire" en $\mathcal{O}(n^3)$

Multiplication de matrices (diviser pour régner)

Multiplication de matrices : Algorithme de Strassen (1969) (biographie)

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} = \begin{pmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{pmatrix}$$

où

$$\begin{aligned}
M_1 &= (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2}), \\
M_2 &= (A_{2,1} + A_{2,2})B_{1,1}, \\
M_3 &= A_{1,1}(B_{1,2} - B_{2,2}), \\
M_4 &= A_{2,2}(B_{2,1} - B_{1,1}), \\
M_5 &= (A_{1,1} + A_{1,2})B_{2,2}, \\
M_6 &= (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2}), \\
M_7 &= (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2}).
\end{aligned}$$

$$C(n) = 7C(\frac{n}{2}) + \Theta(n) \text{ d'où } C(n) = \Theta(n^{\log_2 7}) = \Theta(n^{2.8})$$

réduction de la complexité par rapport à la multiplication "élémentaire" en $\mathcal{O}(n^3)$

RECONNAÎTRE UN SCHÉMA DIVISER POUR RÉGNER

Nuage de n points : trouver la distance minimale entre 2 points

DIVISER POUR RÉGNER

MISE EN ŒUVRE

Adaptation lorsque la taille du problème n'est pas une puissance du facteur de division b

$$C(n) = aC\left(\frac{n}{b}\right) + \text{Coût}_{\text{division/recomposition}}(n)$$

Mise en œuvre

Adaptation lorsque la taille du problème n'est pas une puissance du facteur de division b

$$C(n) = aC\left(\frac{n}{b}\right) + \text{Coût}_{\text{division/recomposition}}(n)$$

Adaptation du programme

- Global : on modifie le programme pour toute taille
- Local : à chaque étape de division on adapte le programme

MISE EN ŒUVRE

Adaptation lorsque la taille du problème n'est pas une puissance du facteur de division b

$$C(n) = aC\left(\frac{n}{b}\right) + \text{Coût}_{\text{division/recomposition}}(n)$$

Adaptation du programme

- Global : on modifie le programme pour toute taille
- Local : à chaque étape de division on adapte le programme

Complexité inchangée

Adaptation des données

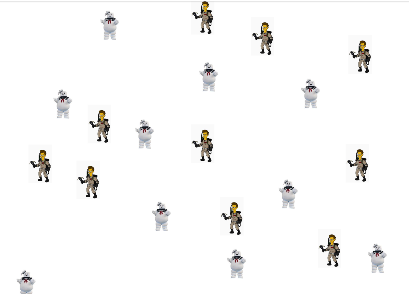
- Global : on étend les données à une puissance de b
- Local : on étend les données à un multiple de b

DIVISER POUR RÉGNER

GHOSTBUSTERS

There's something very important I forgot to tell you ! Don't cross the streams. . . It would be bad. . .Try to imagine all life as you know it stopping instantaneously and every molecule in your body exploding at the speed of light. Dr. Egon Spengler

Les chasseurs de fantômes Ghostbusters utilisent des rayons pour neutraliser les fantômes. Comme l'indique la citation ci-dessus, il est primordial que ces rayons ne se croisent jamais.



Problème : associer chaque Ghostbuster à un fantôme

ANALYSE DU PROBLÈME

Solution bulldozer

Énumération systématique de tous les appariements possibles : $n!$ appariements

- Coût déraisonnable
- on remarque que pour la plupart des exemples on trouve "Solution" intuitive facile
- Énumération "intelligente" ?

Analyse de la faisabilité

- On trouve rapidement des situations sans solution, par exemple lorsque les points sont tous alignés.
- Hypothèse simplificatrice : les points sont en situation générale (pas d'alignements)
- **Théorème** Lorsque les $2n$ points sont en situation générale, il existe un appariement sans croisement de segments

Trouver un algorithme qui résout le problème et prouver cet algorithme permet de prouver le théorème et donne une construction explicite pour toute instance donnée.