Complexité des algorithmes : Motivations [1]

Salem BENFERHAT

Centre de Recherche en Informatique de Lens (CRIL-CNRS)
email : benferhat@cril.fr

[1] Version préliminaire du cours. Tout retour sur la forme comme sur le fond est le bienvenu.

Un algorithme

Intuitivement...

- Il permet de résoudre un problème donné en un certain nombre d'étapes dites "élémentaires"
- Il admet des données en entrée et fournit des résultats en sortie
- il peut-être vu comme comme une procédure "calculatoire" qui transforme des données "entrée" vers des données "sortie"

Qualité d'un algorithme ou d'un programme

On s'intéresse à deux aspects :

Correction et terminaison

- S'assurer que l'algorithme est "Juste" (fais ce que l'on lui demande de faire) et se termine.
- Cours sur les preuves de programmes.

Qualité d'un algorithme ou d'un programme

On s'intéresse à deux aspects :

Correction et terminaison

- S'assurer que l'algorithme est "Juste" (fais ce que l'on lui demande de faire) et se termine.
- Cours sur les preuves de programmes.

Complexité spatiale et surtout temporelle

- Evaluer le temps, ainsi que l'espace mémoire, que prend un algorithme.
- Dire si un problème est faisable ou pas!
- But de ce cours

Exemple

Le programme suivant affiche, de manière aléatoire, un nombre entier compris entre 0 et 9 :

```
2 int main (void)
3 {
4  srand(time(NULL));
5  printf("Voici le nombre aleatoire" : %d , rand()%10);
6  exit(0);
7 }
```

Exemple

Remarques

- Ce programme n'admet pas de données en entrée

Exemple

Remarques

- Ce programme n'admet pas de données en entrée
- De ce fait, deux exécutions de ce programme :
  - sur la même machine,
  - utilisant le même compilateur

  mettront un temps identique pour afficher le résultat.

Un autre exemple

Ecrire un algorithme qui :

- Génère une distribution de probabilité aléatoire. Une façon de le faire est de :
  - générer n nombres aléatoires et les ranger dans un tableau,
  - calculer la somme de ces nombres, et finalement
  - diviser chaque nombre par cette somme
- Calcule l'entropie associée à cette distribution

La fonction qui génère des nombres aléatoires

```
2
3  double *aleatoire (int N)
4  {
5      int i;
6      double *alea=malloc(N*sizeof(double));
7      for (i=0; i<N; i++)
8      {
9          alea[i]=rand()%1000;
0      }
1      return alea;
2  }
```

La fonction Somme

```
2  double Somme(double A[], int N)
3  {
4      double s=0;
5      int i;
6      for (i=0; i< N; i++)
7          s = s + A[i];
8      return s;
9  }
```

La fonction : générer distribution

```
2  double *Dist_probabilites(double A[], int N)
3  {
4      double s;
5      int i;
6      s=Somme(A,N);
7      double *dist=malloc(N*sizeof(double));
8      for (i=0; i< N; i++)
9          dist[i] = A[i]/s;
0      return dist;
   }
```

La fonction : entropie

```
double entropie(double A[], int N)
{
    double resultat=0;
    int i;
    double *dist=malloc(N*sizeof(double));
    dist=Dist_probabilites(A,N);
    for (i=0; i< N; i++)
    {
    if (dist[i]>0)
       resultat=resultat-((dist[i])*log10(dist[i]));
    }
    return resultat;
}
```

Exemple

Remarques

- Contrairement à l'exemple précédent, ce programme admet une donnée en entrée : le nombre de nombres N à générer aléatoirement

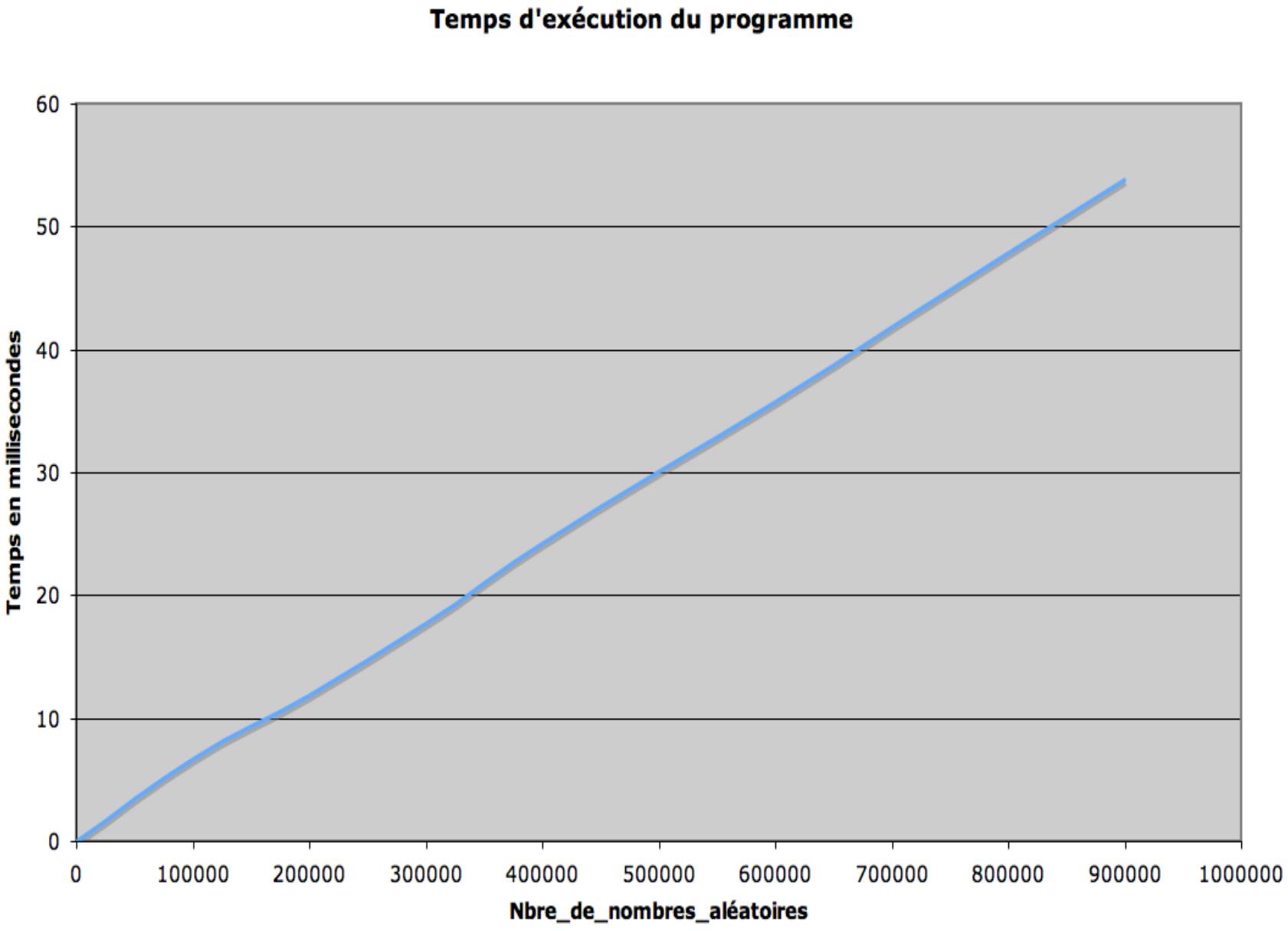
Exemple

Remarques

- Contrairement à l'exemple précédent, ce programme admet une donnée en entrée : le nombre de nombres N à générer aléatoirement
- De ce fait, deux exécutions de ce programme, même :
  - sur la même machine,
  - utilisant le même compilateur

  mettront deux temps différents pour afficher le résultat s'ils utilisent deux entrées (entier N) différentes .

Temps d’exécution



Produit de matrices

Données

Nous disposons de deux matrices NxN de réels

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

et

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

Produit de matrices

Données

Nous disposons de deux matrices NxN de réels

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

et

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

Travail demandé

- Générer aléatoirement deux matrices A et B

Produit de matrices

Données

Nous disposons de deux matrices NxN de réels

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

et

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

Travail demandé

- Générer aléatoirement deux matrices A et B
- Stocker le résultat produit des deux matrices A et B dans une nouvelle matrice C

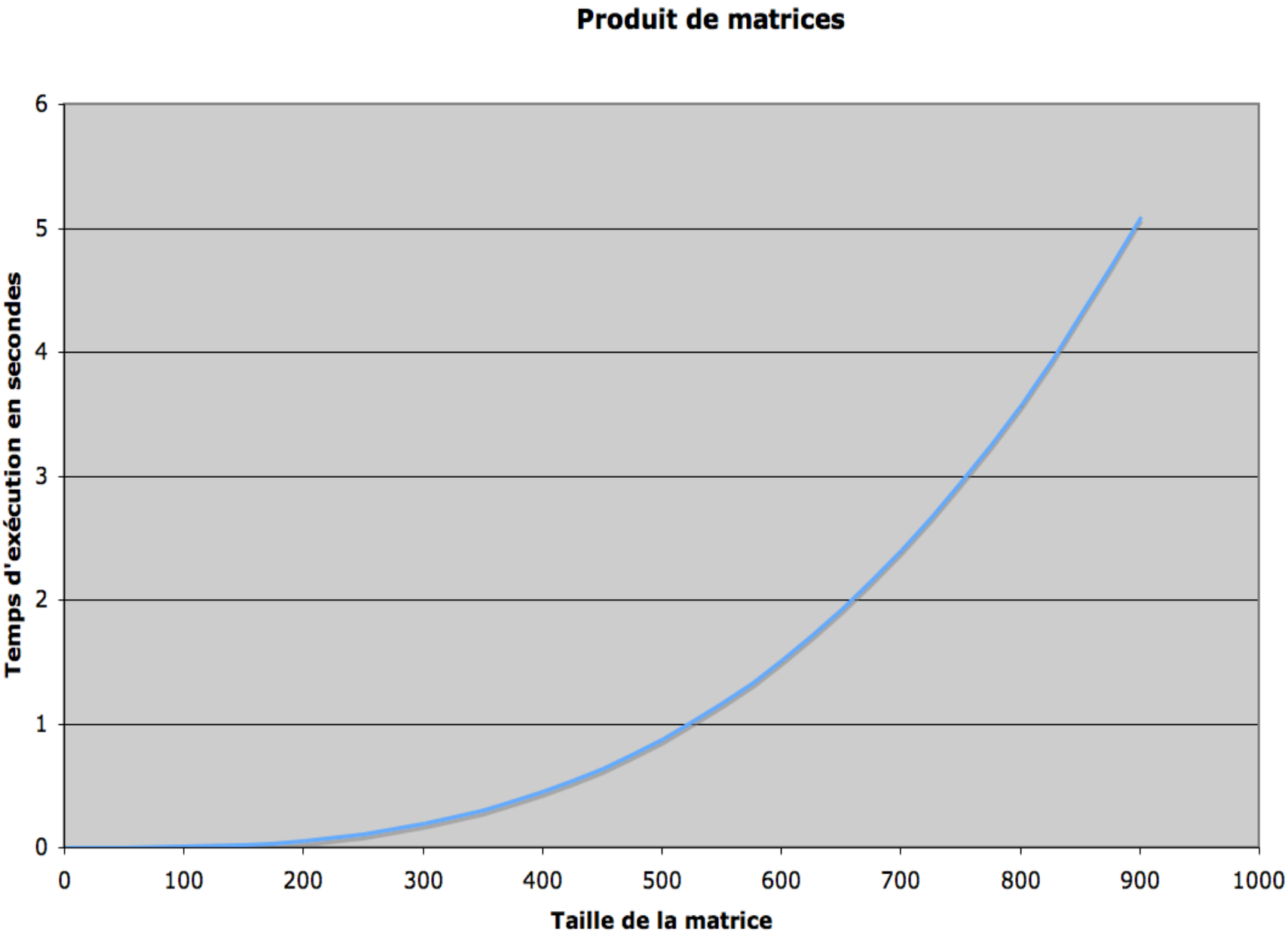
La fonction : génération aléatoire de matrices

```
2
3  double **matrice_aleatoire (int N)
4  {
5     int i,j;
6     double **alea=malloc(N*sizeof(double *));
7     srand(time(NULL));
8     for (i=0; i<N; i++) alea[i]=malloc(N*sizeof(double));
9     for (i=0; i<N; i++)
0     {
1        for (j=0; j<N; j++) alea[i][j]=rand()%10;
2     }
3     return alea;
4  }
```

La fonction : produit de matrices

```
double **produit_matrice(double **A, double **B, int N)
{
    int i,j,k;
    double **C=malloc(N*sizeof(double *));
    for (i=0; i<N; i++) C[i]=malloc(N*sizeof(double));
    for (i=0; i<N; i++)
    {
        for (j=0; j<N; j++)
        {
            C[i][j]=0;
            for (k=0; k<N; k++)
            {
                C[i][j]=C[i][j]+A[i][k]*B[k][j];
            }
        }
    }
     return C;
}
```

Temps d’exécution



Produit de matrices : Remarques

- Pour une entrée N donnée, le calcul du produit de deux matrices est plus long que le calcul de l'entropie.
- En particulier, le calcul de l'entropie se fait en moins d'une seconde pour $N = 10^8$. Ce qui est impossible à exécuter pour le calcul du produit des matrices.

Produit de matrices : Remarques

- Ce cours ne concerne pas l'optimisation du code.
- Par exemple, dans le produit des deux matrices, remplacer la boucle interne (pour le calcul d'un case C[i,j]):

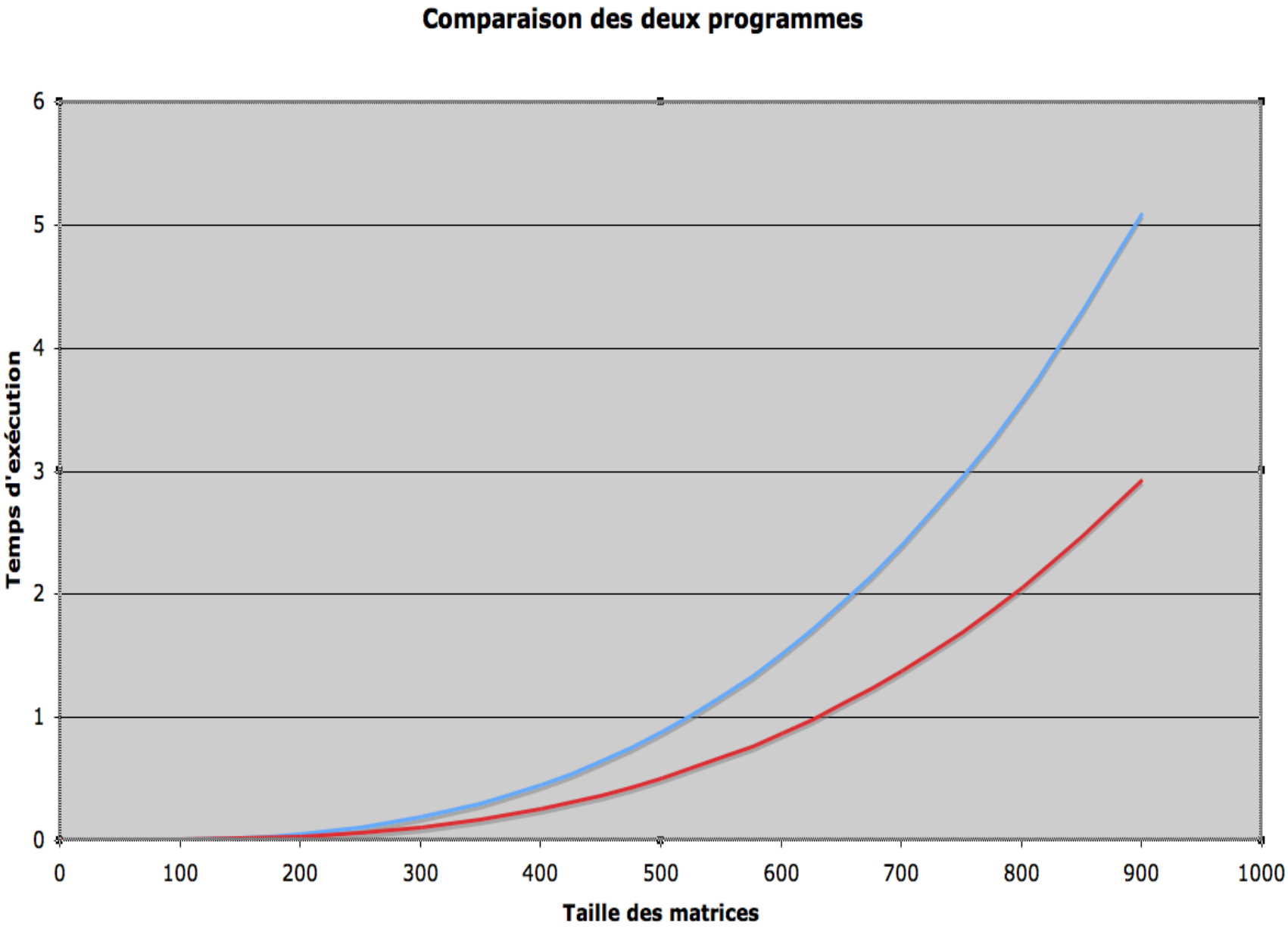
```
C[i][j]=0;
for (k=0; k<N; k++)
   C[i][j]=C[i][j]+A[i][k]*B[k][j];
```

par :

```
s=0;
for (k=0; k<N; k++)
   s=s+A[i][k]*B[k][j];
C[i][j]=s;
```

où s est une variable (de type double).

Temps d’exécution



La fonction : produit de matrices

- Cette petite modification évitera le calcul de la position de ”C[i,j]” en manipulant les pointeurs.
- Elle a permis de réduire le temps de calcul de moitié

Remarques

Dans ces programmes :

- certains n'admettent pas de paramètres,

Remarques

Dans ces programmes :

- certains n'admettent pas de paramètres,
- d'autres ne dépendent que d'un seul paramètre, par exemple :

Remarques

Dans ces programmes :

- certains n'admettent pas de paramètres,
- d'autres ne dépendent que d'un seul paramètre, par exemple :
  - produit des matrices : la taille des matrices carrées,
  - la vérification si un nombre est premier : le nombre à tester.

Remarques

Dans ces programmes :

- certains n'admettent pas de paramètres,
- d'autres ne dépendent que d'un seul paramètre, par exemple :
  - produit des matrices : la taille des matrices carrées,
  - la vérification si un nombre est premier : le nombre à tester.
- d'autres utilisent deux paramètres, par exemple :
  - variante du jeu "pommes, pêches, poires, abricots" : le nombre de joueurs ainsi que le nombre qui sert à éliminer les joueurs,
  - le nombre de nombres premiers entre deux nombres A et B : les deux nombres A et B
- Il est important d'identifier les variables ou les paramètres entrées d'un programme.

Complexité d'un problème vs complexité d'un algorithme

Problèmes = plusieurs algorithmes

Un problème peut-être résolu par plusieurs algorithme.

Complexité d'un problème vs complexité d'un algorithme

Problèmes = plusieurs algorithmes

Un problème peut-être résolu par plusieurs algorithme.

La complexité d'un problème
=
La complexité du meilleur algorithme qui le résout.

Complexité d'un problème vs complexité d'un algorithme

Exemple

- Problème : calculer le nombre de nombres premiers entre a et b.

Complexité d'un problème vs complexité d'un algorithme

Exemple

- Problème : calculer le nombre de nombres premiers entre a et b.
- Différentes manières pour le résoudre, car il existe différents algorithmes pour vérifier si $a \leq p \leq b$ est premier

Complexité d'un problème vs complexité d'un algorithme

Exemple

- Problème : calculer le nombre de nombres premiers entre a et b.
- Différentes manières pour le résoudre, car il existe différents algorithmes pour vérifier si $a \leq p \leq b$ est premier
  - Algorithme 1 : chercher un diviseur de p entre 2 et p.
  - Algorithme 2 : chercher un diviseur de p entre 2 et $\frac{p}{2}$.
  - Algorithme 3 : chercher un diviseur de p entre 2 et $\sqrt{p}$.
  - Algorithme 4 : algorithme approximatif probabiliste.

Complexité d'un problème vs complexité d'un algorithme

Intuitivement

Algorithme 4 (probabiliste) est plus efficace (en terme de temps de calcul) que Algorithme 3 (qui ne va pas au delà de la racine carrée de p), qui est à son tour plus efficace que Algorithme 2 (qui ne va pas au delà de $\frac{p}{2}$) . Algorithme 1 (qui considère tous les cas) est le moins efficace.

Question

Comment confirmer cette intuition? Dis autrement, comment évaluer la complexité temporelle de ces algorithmes?

Comment évaluer la complexité temporelle?

Une manière immédiate de le faire

- Procéder à une étude expérimentale

Comment évaluer la complexité temporelle?

Une manière immédiate de le faire

- Procéder à une étude expérimentale
- Définir un contexte d'expérimentation.

Comment évaluer la complexité temporelle?

Une manière immédiate de le faire

- Procéder à une étude expérimentale
- Définir un contexte d'expérimentation.
- C'est-à-dire choisir d'abord :
  - un langage de programmation pour implémenter ces algorithmes
  - une machine,
  - un compilateur,
  - etc

Comment évaluer la complexité temporelle?

Une manière immédiate de le faire

- Générer des données tests (ou des benchmarks)
  - L'idéal est générer toutes les données (impossible en général)
  - A défaut il faut évaluer les algorithmes sur des données représentatives. En particulier, identifier les données favorables, défavorables à chacun des algorithmes.

Comment évaluer la complexité temporelle?

Une manière immédiate de le faire

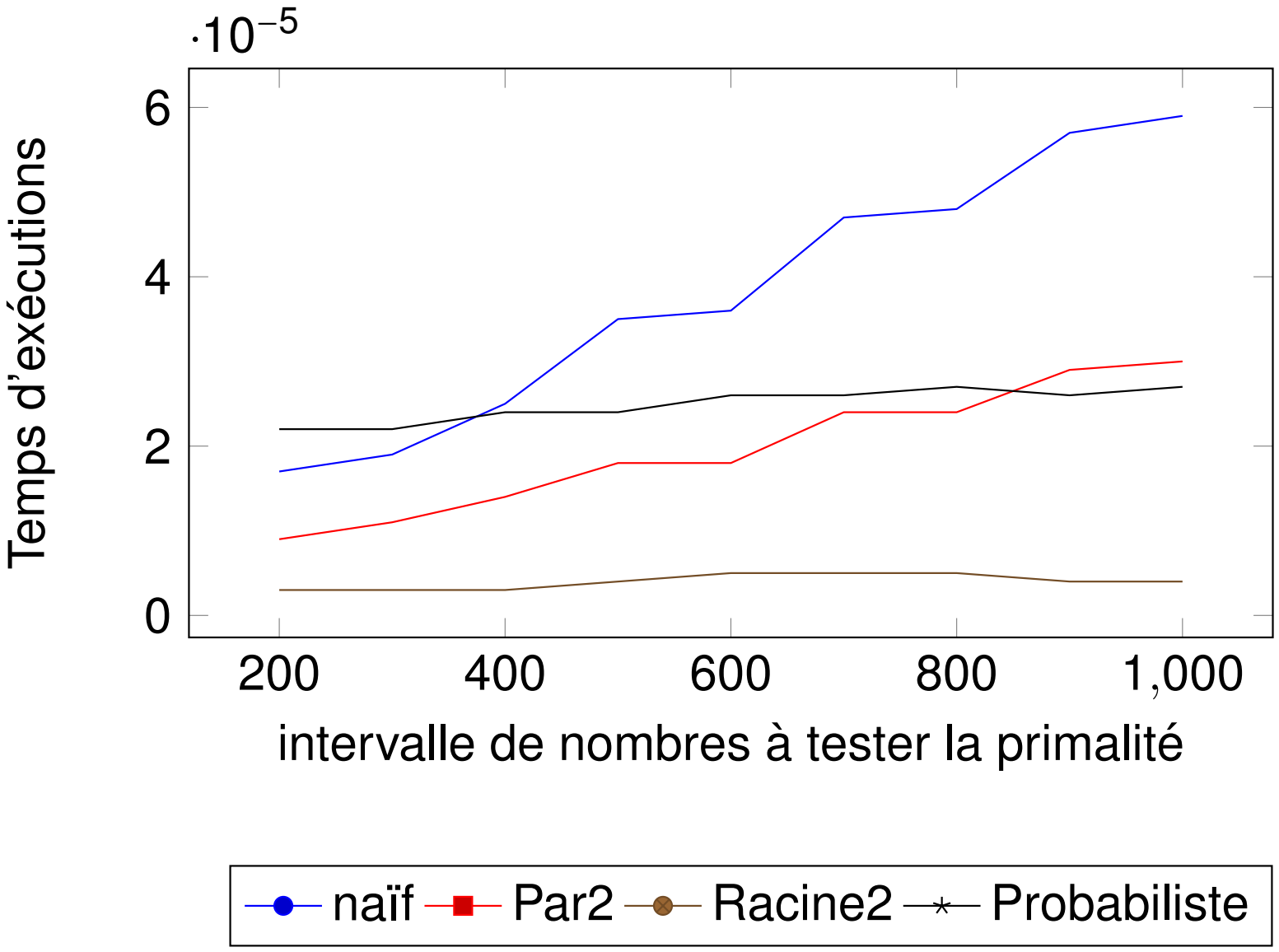
- Dans notre exemple, il faut analyser le comportement des algorithmes sur :
  - des nombres pairs, des nombres impairs
  - des petits nombres (3 à 4 chiffres), des nombres avec une dizaine de chiffres, puis des grands nombres (plusieurs centaines de chiffres)
  - des nombres particuliers ayant par exemple la forme : $2^n - 1$
  - etc

Comment évaluer la complexité temporelle?

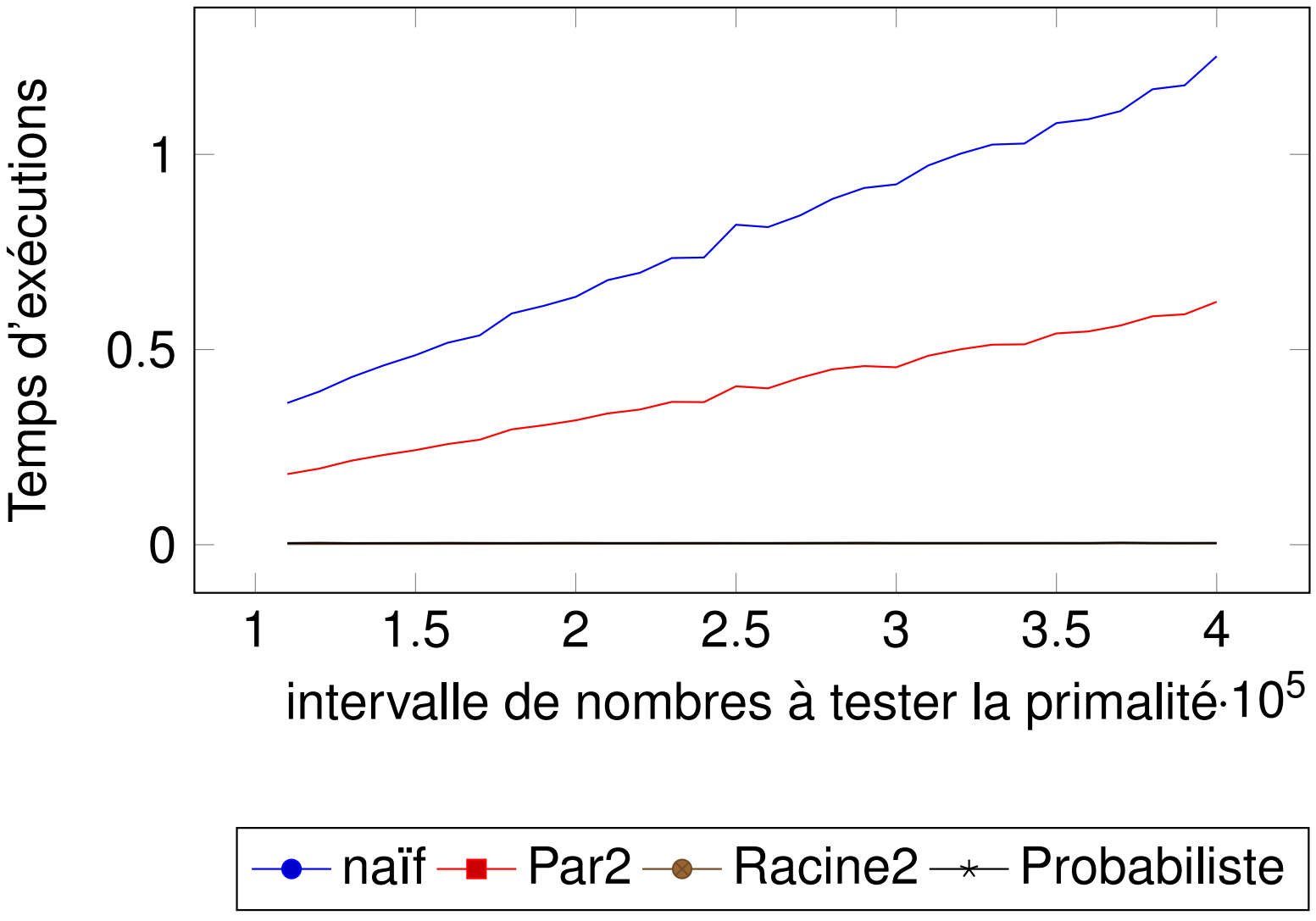
Une manière immédiate de le faire

- Dans notre exemple, il faut analyser le comportement des algorithmes sur :
  - des nombres pairs, des nombres impairs
  - des petits nombres (3 à 4 chiffres), des nombres avec une dizaine de chiffres, puis des grands nombres (plusieurs centaines de chiffres)
  - des nombres particuliers ayant par exemple la forme : $2^n - 1$
  - etc
- Exécuter ces programmes sur ces données
- Récupérer les temps d'exécution, puis comparer ces temps (en traçant les courbes temps d'exécution en fonction des données)

Temps d'exécutions pour les petits nombres (jusqu'à 1000)



Temps d'exécutions pour les petits nombres (jusqu'à 40000)



Problèmes avec l'évaluation empirique

Cette évaluation est nécessaire mais reste insuffisante.

Problèmes avec l'évaluation empirique

Cette évaluation est nécessaire mais reste insuffisante.

Contextes

Les différents temps obtenus dépendent d'un certain contexte :

- Machines
- Langages de programmation
- Compilateurs
- ...

Problèmes avec l'évaluation empirique

Nature des données

- Il n'y a pas que le contexte et la taille des données

Problèmes avec l'évaluation empirique

Nature des données

- Il n'y a pas que le contexte et la taille des données
- La nature des données joue un rôle important.

Problèmes avec l'évaluation empirique

Nature des données

- Il n'y a pas que le contexte et la taille des données
- La nature des données joue un rôle important.
  - Certaines données sont favorables à un algorithme
  - D'autres sont défavorables
  - Comment s'assurer que tous les cas possibles sont couverts par les jeux de test générés (de manière aléatoire ou non)?