UFR IM²AG
UNIVERSITÉ Grenoble Alpes

DU ISN - Algorithmique

Travaux dirigés, séance 1

Coût maximal, coût minimal. Coût moyen (analyse probabiliste)

1 Coût, complexité

Le coût d'une séquence d'instructions dépend en général des valeurs de certaines variables du programme. Le coût maximal (respectivement minimal) est la plus grande (respectivement plus petite) valeur du coût qui puisse **être observée** *lors d'une exécution de cette séquence.*

Exercice 1. Itérations emboîtées

Compter le nombre d'opérations *Schtroumpfer* exécutées par chacun des algorithmes suivants.

1.
```
(1) pour i = 1 à n faire
(2)     pour j = 1 à n faire
(3)         Schtroumpfer(x)
```

2.
```
(1) pour i = 1 à n faire
(2)     pour j = 1 à i faire
(3)         Schtroumpfer(x)
```

3.
```
(1) pour i de 5 à n-5 faire
(2)     pour j de i-5 à i+5 faire
(3)         Schtroumpfer(x)
```

4.
```
(1) pour i = 1 à n faire
(2)     pour j = 1 à i faire
(3)         pour k = 1 à j faire
(4)             Schtroumpfer(x)
```

Exercice 2. Valeurs numériques et ordres de grandeurs

On suppose qu'on travaille sur une machine capable d'effectuer environ un milliard d'opérations par seconde.
Calculer (**sans calculatrice**) le temps nécessaire approximatif pour exécuter des programmes dont les coûts sont donnés ci-dessous, avec des données de différentes tailles en entrée :

↓ Coût de l'algorithme \ Taille des données →	1 000	1 000 000	1 000 000 000
n			
$n \log_2 n$			
$n + 1\,000\,000$			
$\frac{n^2}{1\,000} + 1\,000n$			

Lesquels de ces algorithmes sont utilisables :
- à chaque chargement d'une page web ?
- à chaque démarrage d'une machine ?
- pour produire les plans d'une usine ?

Quelle(s) conclusion(s) plus générale(s) en tirez-vous sur les ordres de grandeurs respectifs de ces coûts ?

Exercice 3. Recherche séquentielle

On étudie un algorithme de recherche séquentielle dans une table. On se place dans le cas où il n'y a pas d'hypothèse sur le fait que la table est ordonnée ni sur la présence de l'élément cherché dans la table.

Il s'agit d'évaluer le nombre de comparaisons entre éléments effectuées lors d'une recherche d'un élément x dans une table T de taille n.
- Spécifier et écrire proprement un tel algorithme.
- Déterminer les cas favorables et défavorables et les nombres de comparaisons correspondants.

Exercice 4. Un algorithme un peu plus élaboré

Données : Un tableau T indicé de 1 à n dont les éléments sont compris entre 1 et n
Résultat : à déterminer
S : un tableau d'entiers indicé de 1 à n
for $i = 1 \dots n$
  $S[i] := 0$
for $i = 1 \dots n$
  $S[T[i]] := S[T[i]] + 1$
$i := 1$
$j := 1$
while $i \leq n$ *et* $j \leq n$
  $k = 0$
  while $k < S[j]$
    $T[i] := j$
    $i := i + 1$
    $k := k + 1$
  $j := j + 1$

- En vous aidant de schémas, déterminer ce que fait ce programme.
- Quel est son coût en fonction de n ?

Un dernier pour la route...

Données : Un entier n
Résultat : Un booléen *ploum*
ploum := $false$
$i := 1$
while $i < n - 1$ *et not ploum*
  $i := i + 1$
  ploum := (n modulo $i = 0$)

- Que calcule cet algorithme ?
- Quel est son coût au pire ?
- Pouvez-vous proposer un algorithme plus efficace pour réaliser la même tâche ?

2 Analyse en moyenne

Le coût moyen d'une séquence d'instructions est la moyenne des valeurs du coût, chacune de ces valeurs étant **pondérée par sa probabilité d'être observée dans les conditions de l'expérience**. *Le calcul du coût moyen (sauf dans le cas trivial où le coût maximal est égal au coût minimal) nécessite donc* **toujours** *de faire des hypothèses (probabilistes) sur les valeurs de certaines variables du programme.*

Exercice 1

```
(1) si a > b alors
(2)    pour i = 1 à n faire
(3)          Schtroumpfer(x)
(4) sinon x := b
```

Évaluer le nombre de schtroumpfages exécutés par cet algorithme (en fonction de a et b) :

- Considérer d'abord les cas favorables (coût minimal) et défavorables (coût maximal).
- Faire l'analyse en moyenne, en prenant pour hypothèse que la probabilité pour que le test $a > b$ soit *vrai* est 1/2.
  Peut-on observer le résultat obtenu, lors d'une exécution particulière de l'algorithme ?
- Faire l'analyse en moyenne, en prenant pour hypothèse que la probabilité pour que le test $a > b$ soit *vrai* est 3/4.

Exercice 2

Soit l'algorithme suivant :

```
(1) pour i de de 1 à n faire
(2)       si T[i] > a alors
(3)          Schtroumpfer(T[i])
```

Évaluer le nombre de schtroumpfages exécutés par cet algorithme :

- Considérer d'abord les cas favorables (coût minimal) et défavorables (coût maximal).
- Faire l'analyse en moyenne, en prenant pour hypothèse que la probabilité pour que le test $T[i] > a$ soit *vrai* est 1/2.

Exercice 3

Déterminer le nombre moyen de comparaisons entre éléments dans l'algorithme de recherche séquentielle.

Exercice 4

```
(1) i := 1
(2) tant que i <= n etpuis T[i] > a faire
(3)          Schtroumpfer(T[i])
(4)          i := i+1
```

Évaluer le nombre de schtroumpfages exécutés par cet algorithme dans les cas favorables, défavorables et en moyenne, en prenant pour hypothèse que la probabilité pour que le test $T[i] > a$ soit *vrai* est 1/2.

Exercice 5

Mêmes questions pour l'algorithme :

```
(1) tant que random > p faire
(2)         Schtroumpfer(x)
```

Exercice 6

```
(1) i := rand(1,n)
(2) tant que T[i] ≠ v faire
(3)     i := rand(1,n)
```

On suppose pour tout cet exercice que l'élément v est présent exactement une fois dans le tableau T. L'opération rand(1,n) tire un entier uniformément dans l'intervalle $[1, n]$.

Évaluer le nombre de comparaisons exécutées par cet algorithme :

- Considérer d'abord les cas favorables (coût minimal) et défavorables (coût maximal).
- Faire l'analyse en moyenne, en prenant pour hypothèse que la probabilité pour que l'élément v soit à l'indice i est $1/n$.