Algo L3 Info
Travaux dirigés 1, séance 3

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Calculs de coûts (2)

RAPPEL *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. Le coût moyen 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.*

Travail réalisé pour aujourd'hui

Étudier la complexité en moyenne du tri par insertion dans le cas général. On se place sous l'hypothèse que, dans le tableau initial, toutes les permutations possibles des n éléments (distincts) sont équiprobables. Rédigez cette question sur la feuille de rendu **(20 minutes)**

D'abord, réécrivons l'algorithme de tri par insertion :
Pour i de 2 à n
j := i
Tant que j¿1 et puis T[j] ¡ T[j-1]
échanger T[j] et Tji-1]
j := j-1

Reformulation de l'hypothèse probabiliste.
Les $n!$ permutations des éléments sont équiprobables. On peut raisonner en supposant que ces éléments sont les entiers de 1 à n (ce qui revient à remplacer les éléments par leur rang dans la séquence triée).
Par conséquent, la valeur initiale de $T[n]$ vaut 1, 2, ... ou n avec la probabilité $1/n$.
D'autre part, les $(n-1)!$ permutation des valeurs initales de $T[1..n-1]$ sont elles aussi équiprobables, et représentent les données possibles pour le tri par insertion de $n-1$ éléments : en effet, si $T[n]$ vaut i, on peut recoder les valeurs 1, 2, ... $i-1$, $i+1$, ... n par 1, 2, ... $n-1$.
Notons $C(n)$ le nombre moyen de comparaisons effectuées pour le tri de n éléments, et $k(n)$ le nombre moyen de comparaison effectuées pour l'insertion de $T[n]$ dans la séquence triée $T[1 \dots n-1]$. $C(n)$ vérifie l'équation de récurrence :

$$C(n) = C(n-1) + k(n)$$

Calculons $k(n)$:
Le nombre de comparaison effectuées pour l'insertion de $T[n] = i$ vaut $n - i + 1$ si i vaut $2, 3, \dots n$, et vaut $n-1$ si $i = 1$. (il y a $n-1$ comparaisons si $i = 1$ ou si $i = 2$).
Par conséquent :

$$k(n) = 1/n \times (1 + 2 + \dots n - 1 + n - 1)$$

C'est-à-dire :

$$k(n) = (n+1)/2 - 1/n$$

Donc on résoud le système :

$$C(n) = C(n-1) + (n+1)/2 - 1/n$$
$$C(1) = 0$$

On obtient

$$C(n) = 1/2 \times (3 + 4 + \dots n + n + 1) - (1/2 + 1/3 + \dots 1/n)$$

C'est-à-dire

$$C(n) = n(n+1)/4 + n/2 - H_n$$

où $H_n = 1 + 1/2 + 1/3 + \ldots 1/n$ est le nombre harmonique, qui est de l'ordre de $log(n)$.

On vérifie que $C(2) = 1$ et $C(3) = 8/3$.

Autres exercices de calcul de coût (40 minutes)

1)

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

Evaluer le nombre d'additions exécutées par cet algorithme dans les cas favorables, défavorables et en moyenne, en prenant pour hypothèse que les différents tests $T[i] > a$ soient indépendants, et que la probabilité que chacun soit *vrai* est 1/2.

- Cas favorable : $T[1] \leq a$, le coût minimal est nul.
- Cas défavorable : tous les éléments du tableau sont supérieurs à a, le coût maximal est n.
- Coût moyen : attention, ici même si les tests sont indépendants on ne peut pas simplement multiplier le coût moyen d'un test par le nombre de tests. En effet, par exemple si T[3]$< a$ on ne fera **jamais** le test pour T[4] !
  La méthode de calcul la plus simple est ici de calculer la probabilité que chaque itération soit effectuée. En effet, pour passer dans la i-ème itération (sans présumer si c'est la dernière ou pas) il faut avoir « réussi » tous les tests précédents, soit une probabilité de $\left(\frac{1}{2}\right)^i$. Comme chaque itération a un coût de 1, le coût moyen est :

$$\sum_{i=1}^{n} \left(\frac{1}{2}\right)^i = \frac{1}{2} \times \frac{1 - \frac{1}{2^n}}{1 - \frac{1}{2}} = 1 - \frac{1}{2^n}$$

2) Mêmes questions pour l'algorithme :

```
(1)  tant que random > 1/2 faire
(2)          x := x+a
```

C'est en fait le même exercice que le précédent, mais sans la limite imposée par n.

- Cas favorable : aucune addition.
- Cas défavorable : le test reste vrai, le programme ne s'arrête jamais !
- Coût moyen : en se basant sur l'exercice précédent, ce coût est $\lim_{n \to +\infty} (1 - \frac{1}{2^n}) = 1$

3)

```
(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 $[1, n]$.

Evaluer le nombre d'opérations 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 $1/n$.

Cela revient encore à l'exercice précédent, mais avec une probabilité $\frac{n-1}{n}$ de réussir le test (si on suppose équiprobables tous les tableaux contenant exactement une fois v).

— Cas favorable : élément v trouvé dès le départ, aucune itération.
— Cas défavorable : on ne trouve jamais v, le programme ne s'arrête jamais.
— Coût moyen :

$$\sum_{i=1}^{k} \left(\frac{n-1}{n}\right)^i = \frac{n-1}{n} \times \frac{1-\left(\frac{n-1}{n}\right)^k}{1-\frac{n-1}{n}} = (n-1)\times(1-\left(\frac{n-1}{n}\right)^k) \quad \text{et} \quad \lim_{k\to+\infty} (n-1)\times(1-\left(\frac{n-1}{n}\right)^k) = n-1$$

4) Mêmes questions pour l'algorithme :

(1) n := 1
(2) tant que random <= $1/n$ faire
(3) n := n+1

Là encore, commençons par déterminer la probabilité que chaque itération soit effectuée. Pour passer dans la i-ème itération il faut avoir « réussi » tous les tests précédents, soit une probabilité de $1 \times \frac{1}{2} \times \frac{1}{3} \times \ldots \frac{1}{i} = \frac{1}{i!}$. Comme chaque itération a un coût de 1, le coût moyen est :

$$\sum_{i=1}^{+\infty} \frac{1}{i!} = e - 1 \quad (\text{où } e \text{ est le nombre tel que } \ln e = 1)$$

Encore un programme ... (à préparer pour la prochaîne fois)

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

— Donnez une spécification la plus précise possible de ce programme.
— Quel est son coût ?
— Donnez un invariant de la boucle permettant d'établir la preuve que ce programme respecte la spécification que vous avez donnée.