Chapitre 11 – Algorithmes probabilistes

## II – Algorithmes de Monte-Carlo

* Le résultat dépend de l’aléatoire et peut être faux

**Définitions**

Faux négatif = résultat faux sur une instance positive

Faux positif = résultat vrai sur une instance négative

**Exemple**

Entrée : une fonction polynomiale de degré d

Sortie : vrai si et seulement si la fonction est constante nulle

Algo :

* On fixe un ensemble fini de taille 2d (par ex )
* On choisit uniformément une de ces valeurs et on évalue la fonction en ce point.
* Si le résultat est nul, on en déduit que la fonction l’est.
* Il n’y a pas de faux négatif et la proba de faux positif

## III – Echantillonnage

1. Permutations

Etant donnée un tableau, le mélanger de manière uniforme.

Mélange de Knuth

void shuffle(int\* tab, int n) {

for (int i = 1; i < n; i += 1) {

int k = rand() % (i + 1);

}

swap(tab, i, k);

}

Prouvons que chaque permutation de est obtenue.

On va montrer par réc sur qu’au -ème tour de boucle, chaque permutation de est équiprobable comme résultat pour les cases du tableau entre et .

alors c'est effectivement garanti

On tire et on échange les cases et

Soit une permutation de , on veut montrer qu’on va l’obtenir avec proba

1er cas : si

alors

2e cas : si

Soit tq

1. Tirage de éléments

On considère un tableau de taille dans lequel on veut sélectionner éléments.

Si on reprend la fonction précédente et qu’on récupère le contenu des premières cases du tableau, c’est suffisant.

Donc .

int\* select(int\* tab, int n, int k) {

int\* res = malloc(k \* sizeof(int));

for (int i = 0; i < k; i += 1) {

res[i] = tab[k];

}

for (int i = k; i < n; i += 1) {

int x = rand() % (i + 1);

if (x < k) {

res[x] = tab[i];

}

}

return res;

}

Cas particulier : si , permet de titrer un élément de manière uniforme.

Exemple : on veut trouver une solution à un jeu avec un backtracking et à chaque étape, on veut choisir un coup au hasard. Ce n’est donc pas confortable de déterminer tous les coups possibles, les compter, et les stocker, à la place :

void select() {

int cpt = 1;

int coup\_a\_renvoyer = premier coup possible;

for (int coup = prochain coup; coup < fin; coup++) {

int x = rand() % (cpt + 1);

cpt += 1;

if (x == 0) {

coup\_a\_renvoyer = coup;

}

}

return coup\_a\_renvoyer;

}

Quelle est la proba de sélectionner le coup  ?