INF7440 : Série d'exercices no. 11

1. Algorithme pour le voyageur de commerce

```
Sequence cheminMin;
real coutCheminMin = INFINI;

procedure etendreChemin( Sequence chemin, int n )
{
  while( longueur(chemin) < n ) {
    int s = int(random(0, n))+1;
    while (dejaVisite(chemin, s)) {
      s = int(random(0, n))+1;
    }
    ajouterEnQueue(chemin, s);
  }
}

procedure trouverCircuitMin( int sommetDepart, int n )
{
  Sequence chemin = creer();
  ajouterEnQueue( chemin, sommetDepart );

  etendreChemin( chemin, n );
  cheminMin = chemin;
  coutCheminMin = coutChemin(chemin) + W[queue(chemin)][tete(chemin)];
}
```

Algorithme 1: Un algorithme pour le voyageur de commerce

L'algorithme 1 présente une façon possible de produire un circuit pour le problème du voyageur de commerce.

a. Expliquez brièvement le fonctionnement de l'algorithme.

b. Pouvez-vous donner un exemple où le résultat produit ne serait pas particulièrement intéressant?

c. De quel type d'algorithme s'agit-il : Glouton ou non? Approximation ou heuristique? Si approximation, pouvez-vous donner une borne? Déterministe ou non déterministe? Si probabiliste, Monte Carlo, Las Vegas ou ni l'un ni l'autre?

2. Algorithme pour le voyageur de commerce (bis)

```
Sequence cheminMin;
real coutCheminMin = INFINI;

procedure etendreChemin( Sequence chemin, int n )
{
  while( longueur(chemin) < n ) {
    int  choix;
    real distance = real(INFINI);
    for [s = 1 to n st ~dejaVisite(chemin, s)] {
      if ( W[queue(chemin)][s] < distance ) {
        distance = W[queue(chemin)][s];
        choix = s;
      }
    }
    ajouterEnQueue(chemin, choix);
  }
}

procedure trouverCircuitMin( int sommetDepart, int n )
{
  Sequence chemin = creer();
  ajouterEnQueue( chemin, sommetDepart );

  etendreChemin( chemin, n );
  cheminMin = chemin;
  coutCheminMin = coutChemin(chemin) + W[queue(chemin)][tete(chemin)];
}
```

Algorithme 2: Un autre algorithme pour le voyageur de commerce

L'algorithme 2 présente une façon de produire un circuit pour le problème du voyageur de commerce géométrique (graphe complet où toutes les distances satisfont l'inégalité du triangle).

a. Expliquez brièvement le fonctionnement de l'algorithme.

b. Quelle est la complexité asymptotique du temps d'exécution de cet algorithme?

c. Pouvez-vous donner un exemple où le résultat produit ne serait pas particulièrement intéressant?

d. De quel type d'algorithme s'agit-il : Glouton ou non? Approximation ou heuristique? Si approximation, pouvez-vous donner une borne? Déterministe ou non déterministe? Si probabiliste, Monte Carlo, Las Vegas ou ni l'un ni l'autre?

3. Algorithme pour le problème du *bin packing*

Le problème : Le problème du *bin packing* est le suivant. On a une série de n items où chaque item a un poids compris entre 0 et 1.0, c'est-à-dire $0.0 < \texttt{items[i]} \leq 1.0$, pour $i = 1, \ldots n$. Les divers items doivent être mis dans des boîtes où la capacité de chaque boîte est d'une unité (1.0). Les items ne sont pas divisibles, c'est-à-dire qu'un item ne peut pas être divisé entre plusieurs boîtes. On cherche à *minimiser* le nombre de boîtes requises pour mettre les n items.

```
procedure empaqueterItems( Sequence items ) returns Sequence boites
{
  boites = creer();
  real boiteCourante = 0.0;
  for [i = 1 to longueur(items)] {
    if ( boiteCourante + element(items, i) <= 1.0 ) {
      boiteCourante += element(items, i)
    } else {
      ajouterEnQueue( boites, boiteCourante );
      boiteCourante = element(items, i)
    }
  }
  ajouterEnQueue( boites, boiteCourante );
}
```

Algorithme 3: Un premier algorithme pour le problème du *bin packing*

L'algorithme 3 présente une première façon de résoudre le problème du *bin packing*.

a. Expliquez brièvement le fonctionnement de l'algorithme.

b. Quelle est la complexité asymptotique du temps d'exécution de cet algorithme?

c. Pouvez-vous donner un exemple où le résultat produit ne serait pas particulièrement intéressant?

d. De quel type d'algorithme s'agit-il : Glouton ou non? Approximation ou heuristique? Si approximation, pouvez-vous donner une borne? Déterministe ou non déterministe? Si probabiliste, Monte Carlo, Las Vegas ou ni l'un ni l'autre?

4. Algorithme pour le problème du *bin packing* (bis)

```
procedure empaqueterItems( Sequence items ) returns Sequence boites
{
  boites = creer();
  for [i = 1 to longueur(items)] {
    int j = 1;
    bool ajoute = false;
    while ( j <= longueur(boites) & ~ajoute ) {
      if ( element(boites, j)+element(items, i) <= 1.0 ) {
        definirElement( boites, j, element(boites, j)+element(items, i) );
        ajoute = true;
      }
      j += 1;
    }
    if (~ajoute) {
      ajouterEnQueue(boites, element(items, i))
    }
  }
}
```

Algorithme 4: Un deuxième algorithme pour le problème du *bin packing*

L'algorithme 4 présente une autre façon de résoudre le problème du *bin packing*.

a. Expliquez brièvement le fonctionnement de l'algorithme.

b. Quelle est la complexité asymptotique du temps d'exécution de cet algorithme?

c. Pouvez-vous donner un exemple où le résultat produit ne serait pas particulièrement intéressant?

d. De quel type d'algorithme s'agit-il : Glouton ou non? Approximation ou heuristique? Si approximation, pouvez-vous donner une borne? Déterministe ou non déterministe? Si probabiliste, Monte Carlo, Las Vegas ou ni l'un ni l'autre?

e. Quel serait l'effet si on modifiait l'algorithme en ajoutant, avant la boucle `for`, un appel à une procédure de tri qui trierait les items en ordre *décroissant* de poids (plus précisément en ordre *non croissant*, si on veut pouvoir traiter des items de même poids)?

5. Algorithme *mystère*

```
procedure partitionner( ref int A[*], int inf, int sup, res int posPivot )
{
  posPivot = int(random(inf, sup+1));
  int pivot = A[posPivot];
  A[inf] :=: A[posPivot];
  posPivot = inf;
  for [i = inf+1 to sup] {
    if (A[i] <= pivot) { A[i] :=: A[++posPivot]; }
  }
  A[posPivot] :=: A[inf];    # Interchange les deux elements.
}

procedure procRec( ref int A[*], int inf, int sup, int k ) returns int r
{
  if (inf == sup) {
    r = A[inf];
  } else {
    int posPivot;
    partitionner( A, inf, sup, posPivot );
    if (k == posPivot) {
      r = A[k];
    } else if (k < posPivot) {
      r = procRec( A, inf, posPivot-1, k );
    } else {
      r = procRec( A, posPivot+1, sup, k );
    }
  }
}
procedure proc( int A[*], int n, int k ) returns int r
{ r = procRec( A, 1, n, k ); }
```

Algorithme 5: Un algorithme mystère

Soit l'algorithme 5.

a. Expliquez ce que fait la procédure `proc`.

b. Quel est, dans le pire cas, le temps d'exécution (asymptotique) de cet algorithme? À quoi correspondrait un tel cas?

c. Quel est, *intuitivement*, le temps d'exécution moyen (temps espéré) de cet algorithme?

d. De quel type d'algorithme s'agit-il : Glouton ou non? Approximation ou heuristique? Si approximation, pouvez-vous donner une borne? Déterministe ou non déterministe? Si probabiliste, Monte Carlo, Las Vegas ou ni l'un ni l'autre?