Devoir Maison d'Algorithmique

Quentin GRUCHET

December 2020

1 Partie Théorique

1.1 Question 1

Si nous avons deux colis de poids P1 et P2 tel que P=P1+P2, cela signifie que nous aurons toujours un colis rempli à son maximum. Ce serait la solution optimal.

1.2 Question 2

Un algo qui remplie deux colis à moins de la moitié de leur capacité serait algorithme vraiment très peu optimal. Il serait incompréhensible d'utiliser un tel algo en situation réelle.

1.3 Question 3

Algorithm 1 Premier arrivé, premier servis

```
Input: colis[] : tableau de taille N. P le poids maximum du colis
Output: res[] : tableau de taille N*N
tmp = 0 //contient la valeur de colis[i] + colis[i+1]
 indexHorizontal = 0 //défini la position horizontal dans res
 indexVertical = 0 //défini la position vertical dans res
 for i de 0 a N do
    if tmp + colis[i] ≤ P then
        tmp += colis[i]
         res[indexVertical][indexHorizontal] ⇐ colis[i]
         indexHorizontal++ //On avance horizontalement pour continuer a remplir le colis
    else
        tmp = 0 //On remet la variable temporaire a 0
         indexVertical++ //On se déplace verticalement dans res
         indexHorizontal = 0 //On revient au début de la ligne dans res
         i– //Permet de ne pas avancer dans tab si le colis est plein
    end
end
return res
```

Algorithm 2 Les plus gros d'abord

```
Input: colis[] : tableau de taille N. P le poids maximum du colis
Output: res[] : tableau de taille N*N
colis ⇐ trieSelection(colis, P)
 res ⇐ premierArrivePremierServis(colis, poids)
 return res
```

1.4 Question 4

Algorithm 3 complementarité

```
Input: colis[] : tableau de taille N. P le poids maximum du colis
Output: res[] : tableau de taille N*N
indexH = 0, indexV = 0
 for i de 0 à N do
    if colis[i] == 0 then
        //si on a deja visité cette case, on passe à l'itération suivante
         continue;
    end
    tmp = 0
     for j de i+1 à N do
        if colis[j] == 0 then
            //si on a deja visité cette case, on passe à l'itération suivante
             continue;
        end
        //Si on regarde le i + 1 element
         if tmp == 0 then
            res[indexV][indexH] = colis[j]; //on met colis[i] dans res
             tmp += colis[j]; //On augmente tmp avec colis[j]
             indexH++; //On se deplace horizontalement dans res
             colis[i] = 0; //On met la case i comme vue
        end
        Si on test plus que i + 1
         if tmp + colis[j] ≤ P then
            res[indexV][indexH] = colis[i];
             indexH++;
             tmp += colis[i];
             colis[j] = 0;
        end
    end
    indexH = 0 //On se replace au début de la ligne dans res
     indexV++ //On se deplace verticalement dans res
end
return res
```

1.5 Question 5

Pour l'exemple (2, 6, 1, 5, 8, 4, 5, 7, 5, 3) et P =9, mes algo ont trouvés :

- Premier arrivé, premier servis : { 2 6 1 }, { 5 }, { 8 }, { 4 5 }, { 7 }, { 5 3 }
- Les plus gros d'abord : { 8 }, { 7 }, { 6 }, { 5 }, { 5 }, { 5 4 }, { 3 2 1 }

- complementarité :{ 8 1 }, { 7 2 }, { 6 3 }, { 5 4 }, { 5 }

1.6 Question 6

Voici pour chaque algo, une entrée où chacun d'entre eux trouvent une solution optimal :

- Premier arrivé, premier servis : (9, 1, 8, 2, 7, 3, 6, 4, 5, 5) avec P = 10.
- Les plus gros d'abord : (19, 16, 13, 9, 8, 12, 14) avec P = 19.

1.7 Question 7

2 Partie Pratique

2.1 Question 1

Voir le code donné en annexe.

2.2 Question 2

Voir le code donné en annexe.

2.3 Question 3

Pour 1000 entrées avec 100 objets de poids entre 1 et 10 et un poids maximal par colis de 10 :

- La moyenne pour le premier algo est : 68.000000
- La moyenne pour le deuxieme algo est : 65.760002
- La moyenne pour l'algo optimal est : 56.854000

Pour 1000 entrées avec 100 objets de poids entre 1 et 1000 et un poids maximal par colis de 1000 :

- La moyenne pour le premier algo est : 72.112000
- La moyenne pour le deuxieme algo est : 59.549999
- La moyenne pour l'algo optimal est : 52.471001