- Fiche de TP4 : Programmation dynamique -

- Choix d'activités valuées -

Le but de cette partie est d'implémenter l'algorithme de choix d'activités valuées vu en cours, celui-ci étant basé sur de la programmation dynamique.
Le fichier *tp4ex1.cc* contient des fonctions de bases de gestion des tableaux d'activités ainsi que les fonctions à compléter.
Une activité est encodé par un type structuré `act` contenant 3 champs : `deb`, sa date de début, `fin`, sa date de fin, et `val` sa valeur. La fonction `GenererActivite` permet de remplir le tableau `activites` soit par un exemple fixé, soit de manière aléatoire.
Les résultats attendus sur l'exemple fixé est donné en bas de page.

- Exercice 1 - Activité précédente -

Observer les fonctions implémenter. Rajouter le code de la fonction `TrierActivites` pour trier les activités par date de fin croissante. On peut utiliser la fonction `swap` directement sur des `struct`.
Implémenter la fonction `CalculDernierPred` qui a pour but de remplir le tableau `predMax` afin de vérifier la condition suivante. Pour chaque activité `i`, l'activité `predMax[i]` est la plus grande activité compatible avec l'activité `i` dans l'ordre calculé précédemment et se terminant strictement avant `activites[i].deb`. Si une telle activité n'existe pas, on gardera `predMax[i]=-1`.

- Exercice 2 - Calcul de la valeur d'une solution optimale, prog. dynamique -

Implémenter la fonction `choixMaxProgD` qui calcule la valeur d'une solution optimale suivant l'algorithme vu en cours. Pour rappel, la valeur `actMax[i]` contient la valeur d'un choix d'activités deux-à-deux com-

```
--------------------------------------------------------------------------------
Activites non tries:
[76,78,10] [12,17,2] [13,15,1] [19,28,8] [12,20,7] [44,45,9] [43,45,5] [1,8,3]

Activites tries par dates de fin croissantes:
[1,8,3] [13,15,1] [12,17,2] [12,20,7] [19,28,8] [44,45,9] [43,45,5] [76,78,10]

Calcul de predMax:
predMax[0]=-1 predMax[1]=0 predMax[2]=0 predMax[3]=0 predMax[4]=2 predMax[5]=4
predMax[6]=4 predMax[7]=6

Valeur d'un choix optimal d'activites (en progdyn): 32

Valeur d'un choix optimal d'activites (en recursif): 32

Tableau d'appartenance a une solution locale max:
sol[0]=1 sol[1]=1 sol[2]=1 sol[3]=1 sol[4]=1 sol[5]=1 sol[6]=0 sol[7]=1

Une solution optimale:
[1,8,3] [12,17,2] [19,28,8] [44,45,9] [76,78,10]
Valeur de cette solution: 32
--------------------------------------------------------------------------------
```

FIGURE 1 – Résultats à obtenir sur l'exemple fixé.

patibles de valeur totale maximale et finissant au pire à la date `activites[i].fin`.

- Exercice 3 - Calcul de la valeur d'une solution optimale, récursivité -

Compléter la fonction `int choixMaxRec(int activites[], int predMax[], int k)` qui renvoie la valeur d'un choix d'activités optimal mais en ne faisant seulement que des appels récursifs (stratégie 'top down'). Pour simplifier l'écriture, on fera retourner 0 par la fonction lorsque le paramètre k vaut -1.
Sur des exemples aléatoires, en faisant varier le nombre d'activités en entrée, comparer les temps d'exécution des fonctions `choixMaxProgD` et `choixMaxRec`. On notera que très rapidement l'approche 'programmation dynamique' est bien plus efficace que le 'calcul récursif' sur ce problème.

- Exercice 4 - Calcul d'une solution optimale -

Pour calculer explicitement une solution de valeur optimale, on utilise un tableau `int sol[]` remplit par l'appel de la fonction `int choixMaxProgDSol(int n, act activites[], int predMax[], int solProgD[], int sol[])`. La variable `sol[i]` contient 1 si, et seulement, si l'activité `i` est la dernière activité d'une sous-solution optimale au problème restreint aux activités se terminant au pire à date `activite[i].fin`.
Enfin la fonction `void ChoixAct(int n, int predMax[], int sol[], int choix[])` remplit le tableau `choix` de telle sorte que l'ensemble des activités `i` avec `choix[i]=1` forme une solution de valeur maximale. Pour remplir le tableau `choix`, on pourra trouver l'activité `i` avec plus grande date de fin et vérifiant `sol[i]=1`, puis l'activté précédente dans une solution optimale et ainsi de suite.

- Voyageur de commerce monotone -