Corrigé d'examen Algorithme 30/5/2007 (ING1)

1. Graphe

Question 1 (10 pt)

Déroulement : 8 pt

	Pi(e)	Pi(a)	Pi(b)	Pi(c)	Pi(d)	Pi(s)	Pivot	S
init	0	Inf	Inf	Inf	Inf	Inf	e	{e}
1	0	3	1	Inf	Inf	Inf	b	{e,b}
2	0	2	1	3	6	Inf	a	{e,b,a}
3	0	2	1	3	6	Inf	c	{e,b,a,c}
4	0	2	1	3	4	8	d	{e,b,a,c,d}
5	0	2	1	3	4	8	s	{e,b,a,c,d,s}

Résultat (2 pt) : le chemin le plus court de e ver s est : e -> b-> c-> s, le coût est 8

Question 2 (5 pt)

Le corrigé (en rouge : les modifications par rapport à l'énoncé**):**

Observateur Graphe algoDijkstra (Entier r, **Reel vmax**) : Graphe

```
// Le paramètre implicite
Observé gr
Références locales
Ensemble s
Entier pivot, nbSommets, j, y, noPere, no
Vecteur pi, pere, succ
Graphe res
Réel val

// Début de l'algorithme
DEBUT
        nbSommets <-- recNbSommets(gr)
        père <- creerVecteur(1,nbSommets)
        j ← 1
        TANTQUE j <= nbSommets FAIRE
                DEBUT
                affVal(père,j,-1)
                j ← j+1
                FIN

        s <-- ensembleVide()
        ajouter(s,r)
        pivot <-- r
```

```
pi <-- creerVecteur(1, nbSommets)
affVal(pi,r,0)
// On met la valuation de tous les sommets autre que r à +infini
j <-- 1
TANTQUE j <= nbSommets FAIRE // Structure itérative
DEBUT
	SI NON estEgal(j,r) ALORS // Structure conditionnelle
	DEBUT
		affVal(pi,j,MAX_INT)
	FIN
	j <-- j + 1
FIN

// On démarre l'algorithme de Dijsktra
j <-- 1
TANTQUE j <= (nbSommets - 1) FAIRE // Structure itérative
DEBUT
	succ <-- recSuccesseurs(gr,pivot)
	no <-- 1
	TANTQUE no <= borneSup(succ) FAIRE // Structure itérative
	DEBUT
		y <-- recVal(succ,no)
		SI NON appartient(s,y) ALORS // Structure conditionnelle
		DEBUT
			SI recval(pi,pivot) + recValuation(recArete(gr,pivot,y)) < recVal(pi,y)   ALORS
			DEBUT
				affVal(pi,y,recval(pi,pivot) + recValuation(recArete(gr,pivot,y)))
				affVal(père, y, pivot)
			FIN
		FIN
		no <-- no + 1
	FIN

	// on cherche le prochain pivot
	y <-- 1
	min <-- MAX_INT
	TANTQUE y <= nbSommets FAIRE // Structure itérative
	DEBUT
		SI NON appartient(s,y) ALORS // Structure conditionnelle
		DEBUT
			SI recVal(pi,y) < min ALORS // Structure conditionnelle
			DEBUT
				ymin <-- y
				min <-- recVal(pi,y)
			FIN
		FIN
		y <-- y + 1
	FIN
	pivot <-- ymin
	ajouter(s,pivot)
	j <-- j + 1
FIN

// On crée le graphe résultat simplement avec les sommets
res <-- creerGraphe(nbSommets)
```

```
        // On crée les arrêts de parcours de plus court chemin dont la valuation est inférieure ou égal  à vmax
        j ← 1
        TANTQUE j <= nbSommets FAIRE
                DEBUT
                val ← recVal(pi,j)
                SI (val <= vmax)  ALORS
                    DEBUT
                    noPere ← recVal(père, j)
                    SI non estEgal(noPere,-1) ALORS
                            DEBUT
                            ajouterArete(res, créerAreteValuee(noPere,j,recValuation(recArete(gr,noPere,j))))
                            FIN
                    FIN
                j ← j+1
                FIN

        retourner res
FIN
```

2. Complexité

Question 1 (5 pt)

```
void trier(int [] tableau, int n)
{
        int pas, i, j, memoire;
        pas = 0;
        // Calcul du pas
        while(pas < n)                                          // Cn=3Cn-1 + 1 => complexité log3n
        {                                                       // opérations : tc (<), 2top (*,+), t(=)
                pas = 3*pas+1;
        }
        while(pas!=0) // tant que le pas est > 0                 // log3n fois
        {
                pas = pas/3;
                for(i=pas; i < n; i++)
                {
                        memoire = tableau[i]; // valeur à décaler éventuellement
                        j = i;
                        while((j > (pas-1)) and (tableau[j-pas] > memoire))
                        {       // échange des valeurs
                                tableau[j] = tableau[j-pas];
                                j = j-pas;
                        }
                        tableau[j] = memoire;
                }
        }
}
```

…