

informatique / licence 3

## 2. Écrire une fonction `print_paths` de signature

```
void print_paths(graphe_t *g, int sommet, int *pred);
```

affichant les trajets minimaux entre le sommet de départ et tous les sommets du graphe sous la forme suivante :

```
S0 -> S0 : S0
S0 -> S1 : S0 S5 S1
S0 -> S2 : S0 S5 S1 S6 S11 S2
S0 -> S3 : S0 S5 S1 S6 S11 S2 S12 S7 S17 S3
S0 -> S4 : S0 S5 S1 S4
S0 -> S5 : S0 S5
etc.
```

Une fonction auxiliaire (récursive) pourra être utilisée.

## 3. Écrire une fonction `write_graphviz2` de signature

```
void write_graphviz2(FILE *f, graphe_t graphe, double *dist, int *pred);
```

produisant un graphique similaire à celui de la page précédente.

Le début du fichier (en langage dot) sera :

```
digraph G {
  layout=sfdp
  splines=curved
  node [
    colorscheme=dark28
    penwidth=1.75
    shape=circle
    fixedsize=true
  ]
  edge [
    colorscheme=dark28
    penwidth=1.75
    arrowhead=normal
  ]
}
```

Le sommet de départ sera déclaré de la façon suivante :

```
S0 [ color=2, style=filled, fillcolor=2 ]
```

(ce sommet se reconnaît car il est le seul à avoir une valeur égale à -1 dans le tableau des prédécesseurs `pred`) — les autres sommets ne seront pas déclarés.

Les arêtes seront tracées avec leur poids et leur couleur de la façon suivante :

```
S1 -> S11 [label=7, color=2]
S2 -> S7 [label=23, color=8]
```

où la couleur 8 est utilisée pour les arêtes inutilisées et la couleur 2 pour les arêtes utilisées dans les plus courts chemins.