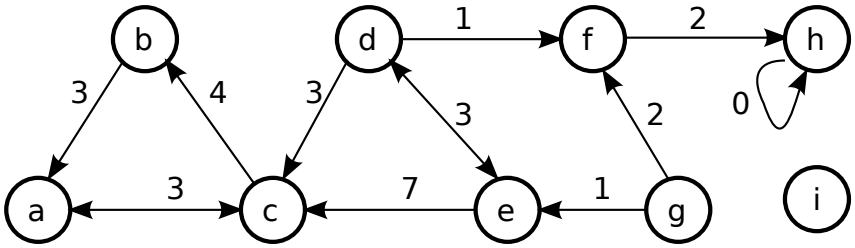
/**** Quiz 2 / Annexe ****/

1 Monceau (*heap*)

Pour les questions portant sur la structure monceau, vous devez supposez que le monceau garde l'élément le plus petit (et non le plus grand) à la racine.

2 Simulation d'algorithmes sur le graphe A

Le graphe A est le suivant.



Pour les questions demandant de simuler les algorithmes de recherche en profondeur et en largeur dans le graphe A, supposez :

1. Les arêtes sortantes sont énumérées en ordre alphabétique. Par exemple, les arêtes sortantes du sommet d sont énumérérés dans l'ordre $\langle c, e, f \rangle$.
2. Un sommet est réputé visité au moment où il est marqué visité. Cela correspond à la ligne 2 de l'algorithme 1 et à la ligne 9 de l'algorithme 2. Cela correspond aussi au moment d'affichage (`cout << ...`) dans le code source des fonctions `GrapheX::rechercheProfondeur` et `GrapheX::rechercheLargeur` fournies au verso.

Rappels des algorithmes.

Algorithme 1 Recherche en profondeur

```
1.  RECHERCHEPROFONDEUR(G = (V,E), v ∈ V)
2.      v.visité ← vrai
3.      pour toute arête e ∈ v.aretesSortantes()
4.          w ← e.arrivee
5.          si ¬w.visité
6.              RechercheProfondeur(G, w)
```

Algorithme 2 Recherche en largeur

```
1.  RECHERCHELARGEUR(G = (V,E), v ∈ V)
2.      file ← CRÉERFILE
3.      s.visité ← vrai
4.      file.ENFILER(v)
5.      tant que ¬file.vide()
6.          s ← file.defiler()
7.          pour tout arête a = (s,s') ∈ E
8.              si ¬s'.visité
9.                  s'.visité ← vrai
10.                 file.ENFILER(s')
```

3 Analyse de la complexité du code au verso

Supposez : n = nombre de sommets ; m = nombre d'arêtes ; et $n < m < n^2$. Notez qu'il y a au plus une arête sortante partant d'un sommet x vers un sommet y. Soyez conscients que la représentation des classes et des fonctions au verso ne sont pas forcément optimales. Ainsi, leur complexité peut être supérieure au résultat de l'analyse présentée en classe et dans les notes de cours.

```cpp
// Représentation de graphe 1
class Graphe1 {
    struct Arete{
       string depart, arrivee;
       double etiquette;
    };
    set<string> sommets;
    mutable set<string> visites;
    list<Arete> aretes;
    //...
};
```

```cpp
// Représentation de graphe 2
class Graphe2 {
    Graphe2();
    ~Graphe2();
    vector<string> sommets; // noms des sommets
    map<string, int> index; // nom -> indice
    bool** matrice; // matrice[s1][s2]=true
    indique une arête de s1 à s2
    mutable vector<bool> visites;
    //...
};
```

```cpp
void Graphe1::parcoursRechercheProfondeur(const string& s) const{
    visites.clear();
    parcoursRechercheProfondeur2(s);
}
void Graphe1::parcoursRechercheProfondeur2(const string& s) const{
    if(visites.find(s)==visites.end()) return;
    visites.insert(s); cout << s << '\t';
    for(list<Arete>::const_iterator i=aretes.begin();iter!=aretes.end();++iter)
        if(s==(*iter).depart) parcoursRechercheProfondeur2((*iter).arrive);
}
void Graphe1::parcoursRechercheLargueur(const string& s) const{
    visites.clear();
    queue<string> file;
    visites.insert(s); cout << s << '\t';
    file.push(s);
    while(!file.empty()){
        string suivant = file.front();
        file.pop();
        for(list<Arete>::const_iterator i=aretes.begin();iter!=aretes.end();++iter)
            if(suivant==(*iter).depart && visites.find((*iter).depart)==visites.end()){
                visites.insert((*iter).arrivee); cout << (*iter).arrivee << '\t';
                file.push((*iter).arrivee);
            }
    }
}
void Graphe2::parcoursRechercheProfondeur(const string& s) const{
    for(int i=0;i<visites.size();i++) visites[i]=false;
    parcoursRechercheProfondeur2(index[s]);
}
void Graphe2::parcoursRechercheProfondeur2(int indice) const{
    if(visites[indice]) return;
    visites[indice]=true;  cout << s << " ";
    for(int j=0;j<sommets.size();j++)
        if(matrice[indice][j])
            parcoursRechercheProfondeur2(j);
}
void Graphe2::parcoursRechercheLargueur(const string& s) const{
  /* La meilleure implémentation possible sans changer la représentation de Graphe2 */
}
```