PRÁCTICA III: ALGORITMOS GREEDY

BY: ANTONIO GÁMIZ DELGADO Y ELENA MERELO MOLINA

Problema

Dado un conjunto de ciudades y una matriz con las distancias entre todas ellas, un viajante debe recorrer todas las ciudades exactamente una vez, regresando al punto de partida, de forma tal que la distancia recorrida sea mínima. Más formalmente, dado un grafo G, conexo y ponderado, se trata de hallar el ciclo hamiltoniano de mínimo peso de ese grafo.

Problema del Viajante de Comercio (solución 1)

```
int graph::nearest_city(int i, double &min_dist){
  int j, n= cities.size();
  set<pair<double, int> > posibilidades;
  set<pair<double, int> >::iterator it;

  min_dist= 0;

  for(j= 0; j< n; j++){
    //Si estamos en el triángulo superior de la matriz de adyacencia
    if(!visited[j]){
      if( i > j)
        posibilidades.insert(make_pair(m[j][i], j));  //insertamos la distancia entre las ciudades y a qué ciudad va

      //Si no está en el triángulo superior obtenemos la coordenada simétrica
      else if( i< j)
        posibilidades.insert(make_pair(m[i][j], j));

      //Si i == j no se hace nada.
    }
  }

  //Como el set ordena automáticamente sus componentes, en la primera posición estará la mínima distancia
  it= posibilidades.begin();
  min_dist= it->first;
  return it->second;
}
```

Problema del viajante de comercio (solución 1)

```
vector<int> graph::min_path1(int i, double &l){
  int n= cities.size(), j;
  l= 0;
  assert( i >= 0 && i < n);

  double min_dist;
  vector<int> r;

  //Mientras haya ciudades por recorrer
  while(!finished_path()){
    //Añadimos la ciudad destino a la lista de ciudades recorridas
    r.push_back(i);
    visited[i]= true;

    j= nearest_city(i, min_dist);
    //Sumamos la distancia a la cantidad de camino recorrido
    l += min_dist;

    //Nos situamos en la ciudad destino y buscamos desde ahí el mínimo camino
    i= j;
  }
  close_path(r, l);

  return r;
}
```

Problema del Viajante de Comercio (solución 2)

```cpp
int graph::westernmost_city(){
  double x=LONG_MAX;
  int index;

  for(int i=0; i<cities.size(); i++)
    if( cities[i].first < x ){
      x=cities[i].first;
      index=i;
    }
  return index;
}

int graph::easternmost_city(){
  double x=-LONG_MAX;
  int index;
  for(int i=0; i<cities.size(); i++)
    if( cities[i].first > x ){
      x=cities[i].first;
      index=i;
    }
  return index;
}
```

```cpp
int graph::northernmost_city(){
  double x=-LONG_MAX;
  int index;
  for(int i=0; i<cities.size(); i++)
    if( cities[i].second > x ){
      x=cities[i].second;
      index=i;
    }
  return index;
}


double graph::total_weight(vector<int> path){
  double l=0;
  for(int i=0; i<path.size()-1; i++){
    l+=get_weight(path[i], path[i+1]);
  }
  //Para cerrar el camino
  close_path(path, l);
  return l;
}
```

```cpp
pair<int, double> graph::particular_min(vector<int>& path, int x){
  double aux=0, min=LONG_MAX;
  int index;

  for(int i=0; i<=path.size(); i++){
    path.insert(path.begin()+i, x);

    aux=total_weight(path);
    if( aux<min ){
      index=i;
      min=aux;
    }

    path.erase(path.begin()+i);
  }

  return make_pair(index, min);
}
```

```cpp
pair<int, int> graph::general_min(vector<int> &r){

  set<pair<double,pair<int, int> > >posibilities;
  pair<int, double> aux;

  for(int i=0; i<cities.size(); i++){
    if( !visited[i] ){
      aux=particular_min(r, i);
      posibilities.insert( pair<double,pair<int, int> >( aux.second, make_pair(i, aux.first) ) );
    }
  }

  return make_pair( (*posibilities.begin()).second.first, (*posibilities.begin()).second.second );

}
```

Problema del Viajante de Comercio (solución 2)

```
vector<int> graph::min_path2(double &l){
  clear();
  vector<int> r;

  r.push_back( westernmost_city() );
  r.push_back( easternmost_city() );
  r.push_back( northernmost_city() );

  visited[r[0]]=true; visited[r[1]]=true; visited[r[2]]=true;

  pair<int, int> new_node;
  while( r.size() < cities.size() ){
    new_node = general_min(r);
    visited[new_node.first]=true;
    r.insert(r.begin()+new_node.second, new_node.first);
  }

  l=total_weight(r);
  return r;
}
```

Problema del Viajante de Comercio (solución 3)

```
vector<int> graph::min_path3(int i, double &l){
  clear();
  vector<int> r;

  set<pair<double, int> > posibilities;
  for(int j=0; j<cities.size(); j++)
    if(i!=j) posibilities.insert(make_pair( get_weight(i,j) , j) );

  r.push_back(i);
  r.push_back( (*posibilities.begin()).second );
  r.push_back( (*(++posibilities.begin())).second );

  visited[r[0]]=true; visited[r[1]]=true; visited[r[2]]=true;

  pair<int, int> new_node;
  while( r.size() < cities.size() ){
    new_node = general_min(r);
    visited[new_node.first]=true;
    r.insert(r.begin()+new_node.second, new_node.first);
  }

  l=total_weight(r);
  return r;
}
```

FIN