UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
ENGENHARIA DE COMPUTAÇÃO

Estrutura de Dados II

Agrupamento de espaçamento máximo utilizando uma MST

Trabalho Prático I

Mirelly Micaella da Silva

Professor: Giovanni Ventorim Comarela

Vitória - ES
2019

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
ENGENHARIA DE COMPUTAÇÃO

Introdução

O objetivo deste trabalho é resolver o problema de agrupamento fornecido na especificação que consiste em encontrar grupos naturais em bases de dados. Os objetos de um grupo devem ser similares e objetos de grupos diferentes devem ser não similares.

O problema do agrupamento é extremamente subjetivo, uma vez que o conceito de similaridade pode variar significativamente dependendo do domínio sendo estudado. Tendo isso, para este trabalho foi proposto o Agrupamento de Espaçamento Máximo e para solucioná-lo foi sugerido o uso do Algoritmo de Kruskal .

Foi implementada uma adaptação do algoritmo de Kruskal para obter árvores que formam os conjuntos do agrupamento de espaço máximo.

Neste documento explico mais detalhadamente o processo para a resolução do problema bem como seus resultados, que acabaram não sendo o esperado para alguns casos de teste.

Vitória - ES
2019

Metodologia

Algoritmos

O algoritmo utilizado para encontrar foi o descrito na seção 2.2 da especificação do trabalho bem como o algoritmo para encontrar a árvore geradora mínima (minimum spanning tree - MST), o Algoritmo de Kruskal.

Também pensei em utilizar o Union Find visto em aula exatamente como estava, mas depois percebi que teria que mudar mesmo para adaptá-lo ao algoritmo de Kruskal.

Estruturas

Para facilitar o manuseio e manter a legibilidade do código decidi armazenar os pontos numa struct Point;

```
struct point {
  char *id;
  float *coords;
  int m;
};
```

Também uma struct List, apenas para facilitar a inserção e impressão do que fosse necessário imprimir, deixar mais legível o código e não ficar me preocupando tanto com índices. Essa estrutura foi muito útil na fase de desenvolvimento quando precisei imprimir várias coisas para testar.

```
struct list {
  void **items;
  int size;
  int type;
};
```

Para meu Union Find defini algumas estruturas que facilitariam o uso do meu Algoritmo de Kruskal.

```
struct edge {
```

```
  int src;
  int dest;
  float weight;
};

struct graph {
  int V;      // número de vértices
  int E;      // número de arestas
  List *edges; // lista de arestas
};

struct subset {
  int parent;
  int rank;
};
```

Para armazenar as distâncias fiz a matriz Y proposta na especificação onde Y(i, j) = d(xi , xj). “Podemos pensar que a matriz Y descreve um grafo G completo, não-direcionado e ponderado cujo conjunto de vértices é V = {1, . . . , n}, onde o peso de uma aresta {i, j} é Y(i, j). “

Analise de complexidade

Leitura

O programa le linhas do arquivo até elas acabarem usando as funções strtok() e getline() .

- getline() foi usada 1 vez para cada linhas que foram N no total;
- strtok() foi usada 2 + dimensão do espaço para cada N, então foi m + 2 no total;

Assim, considerando que tivemos N * (2 + m) usos das funções, a complexidade é de O(N*m)

Calculo de distancias

Foi usada uma fórmula para obter uma matriz com a distância euclidiana entre dois pontos que ficou assim:

```
float distance(Point *x, Point *y) {
   float distance = 0;
   for (int i = 0; i < x->m; i++)
       distance += pow(y->coords[i] - x->coords[i], 2);
   return sqrt(distance);
}
```

Em resumo a complexidade é da ordem de $O(n^2)$ porque eu calculo a distância de cada ponto com todos os outros pontos. Pensando bem, metade desses cálculos poderiam ser evitados, mas não há mais tempo. E também, para cada ponto é feito um loop percorrendo suas coordenada, então dependo da dimensão m, temos a complexidade $O(n^2*m)$

Ordenação das distâncias

Considerando que complexidade do qsort é O(N*log N) e o vetor de distâncias possui n^2 elementos, a ordem de complexidade da ordenação das distâncias, em teoria, é $O(n^2 \log n)$.

Gerar MST

Com o algoritmo usado, a complexidade da geração da MST é $O(n^2)$.

Identificar Grupos

Para identificar grupos foi feito apenas um loop com uma quantidade de iterações igual ao número de pontos, então a complexidade é O(n);

Escrita no arquivo

Para cada grupo k vão ser impressos n pontos, então a complexidade é O(n+k) ou simplesmente. O(n)

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
ENGENHARIA DE COMPUTAÇÃO

Analise empírica

Em construção...

Teste	Leitura	Calculo Distancia	Ordenação das Distancias	Gerar MST	Identificar Grupos	Escrita no arquivo
1						
2						
3						
4						
5						

Vitória - ES
2019