Relatório de Análise e Síntese de Algoritmos
2º Projeto

Introdução

Neste relatório, iremos abordar a solução encontrada para o problema proposto pelo corpo docente.

O problema deste projeto consiste na criação de um programa que permita determinar a segmentação de imagens obtidas por uma câmera, isto é, classificar os pixels da imagem como sendo de primeiro plano ou de cenário.

O input dado permite saber o tamanho da imagem, os pesos dos pixeis de primeiro plano e de cenário e os pesos das relações de vizinhança horizontais e verticais de cada pixel.

Através deste input o programa tem de determinar a segmentação que minimiza o peso total, ou seja, a menor soma entre os pesos de primeiro plano, os pesos de cenário e o peso das ligações que existem entre ambos.

Descrição da solução

O programa realizado foi implementado em linguagem C. Para solucionar o problema, a imagem foi interpretada como um grafo e cada pixel como um vértice desse mesmo grafo.

1. Criou-se um grafo através do input com:
    - o número de vértices igual ao número de píxeis da imagem mais dois (source (vértice de origem) e o sink (vértice de destino));
    - o peso das arestas de source para os restantes vértices, igual à posição da matriz de pesos de primeiro plano correspondente;
    - o peso das arestas dos vértices para sink, igual à posição da matriz de pesos de cenário correspondente;
    - o peso das arestas que representam as ligações de vizinhança entre os vértices igual às posições correspondentes das matrizes de relações de vizinhança horizontais e verticais.

2. Para diminuir o número de ligações percorridas pelo algoritmo implementado, na inserção, os pesos de primeiro plano e de cenário são guardados, em cada index respectivo, na lista de nodes em *struct graph*. Mais tarde com o auxílio da função *void vertixType(Graph graph, int t)* compara-se o peso de primeiro plano e o peso de cenário para determinar qual o mais pequeno sendo este o fluxo mínimo do caminho entre a source para o vértice e do vértice para o sink, sendo que obrigatoriamente uma das ligações irá ficar com o fluxo igual à capacidade e não ser percorrida

João Antunes, 87668
Viviana Bernardo, 87709

pelo algoritmo implementado (Edmonds-Karp). A diferença entre os 2 pesos irá ser incrementado ao valor final de peso total.

3. Implementou-se o algoritmo de Edmonds-Karp para descobrir o corte mínimo, ou seja, a segmentação que minimiza o tempo total.

Na nossa versão do algoritmo de Edmonds-Karp temos, como seria de esperar, um ciclo while que é executado enquanto ainda existirem caminhos possíveis no grafo do vértice de origem para o vértice de destino, o que é confirmado usando uma BFS.

No entanto, após esse ciclo terminar, e os valores de fluxo da rede residual estarem atualizados, usamos uma DFS para encontrar o corte mínimo:

- Se o vértice tiver sido visitado pela DFS então assumimos que ele é de cenário, senão fica como vértice de primeiro plano.
- Após fazermos a DFS calculamos também o peso total da segmentação para que inclua os valores dos pesos dos vértices que ficam de primeiro plano, de cenário e das arestas que ligam vértices de primeiro plano a vértices de cenário.

4. Finalmente, imprimiu-se o peso total da segmentação e a matriz que representa a imagem, com cada posição associada a um vértice de cenário ou vértice de plano.

Análise Teórica

Sendo V o número de vértices e E o número de arestas, como o grafo será um grafo esparso então podemos inferir que $|E| \approx 5|V|$.

O algoritmo de Edmonds-Karp implementado tem complexidade $O(V E^2)$, ou seja, $O(V^3)$.

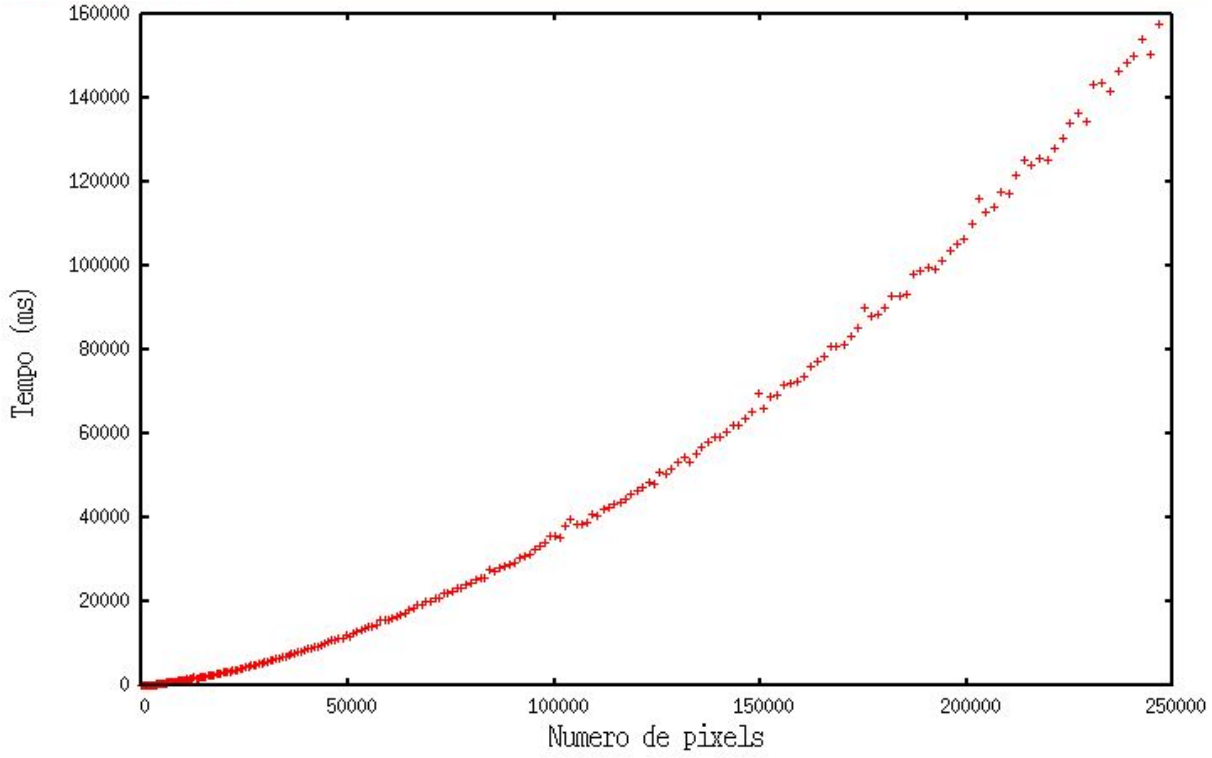
A BFS e DFS implementadas têm ambas complexidade $O(V+E) \approx O(2V) \approx O(V)$.

Portanto, concluímos assim que a nossa complexidade em tempo de execução no pior caso é de $O(V^3) + O(V) + O(V)$ ou seja, $O(V^3)$.

Quanto à complexidade de alocação de memória, esta será $O(V + E) \approx O(V)$ para todos os vértices e ligações guardadas.

Análise Experimental

Gráfico da complexidade em função do tempo*



* - Os valores deste gráfico foram obtidos através de um gerador de testes automático fornecido pelo professor, com esses valores gerámos o respetivo gráfico com auxílio do programa "*gnuplot*".

Podemos confirmar então que o tempo de execução aumenta cubicamente em relação ao número de pixels, tal como o previsto na avaliação teórica.

Referências

- Diapositivos das aulas teóricas de ASA sobre algoritmos elementares em grafos (DFS e BFS) e fluxos máximos (Ford-Fulkerson e Edmonds-Karp):
  https://fenix.tecnico.ulisboa.pt/downloadFile/1407993358870184/ch22.pdf
  https://fenix.tecnico.ulisboa.pt/downloadFile/1407993358870185/ch22_2.pdf
  https://fenix.tecnico.ulisboa.pt/downloadFile/845043405462220/ch26.pdf

- Diapositivos das aulas teóricas de IAED sobre listas ligadas e grafos:
  https://fenix.tecnico.ulisboa.pt/downloadFile/282093452038711/iaed2016_17-2s-aula17-listasInteiros_listasStrings.pdf
  https://fenix.tecnico.ulisboa.pt/downloadFile/1970943312290820/iaed2016_17-2s-aula21-GrafosIeII.pdf

- https://www.geeksforgeeks.org/minimum-cut-in-a-directed-graph/