Instituto Superior Técnico

Análise e Síntese de Algoritmos

Relatório 2º Projeto

Grupo 94 (Alameda) | João Almeida - 86447 | Maria Forjó - 86475

Introdução

O problema proposto baseia-se na segmentação dos píxeis de uma imagem e posterior classificação dos mesmos como sendo de primeiro plano ou de cenário.

O objetivo será desenvolver um algoritmo eficiente, que recebendo os pesos e respetivas relações de vizinhanças entre os píxeis da imagem, determine a segmentação que minimiza o peso total da mesma. O sistema desenvolvido recebe do *standard input* a informação necessária para saber quantos píxeis tem a imagem, e quais as ligações e respetivos pesos. Apresenta no *standard output* o peso total mínimo da segmentação para a figura, tal como uma sequência que indica quais os píxeis de primeiro plano (*P*) e de cenário (*C*). No caso de existirem múltiplas segmentações possíveis, será apresentada a que maximiza o número de píxeis de primeiro plano.

Descrição da Solução

Os píxeis da imagem podem ser vistos como vértices que fazem parte de um grafo não dirigido, onde cada ligação entre eles e o seu peso são, respetivamente, arcos e capacidades.

Foram criados dois vértices artificiais, *source* e *target*. Ambos ligam-se a todos os vértices do grafo, sendo que a capacidade da aresta que liga a *source* a cada pixel é o valor do seu peso de plano, e a da que liga o pixel ao *target* é o valor do seu peso de cenário. Para minimizar a soma do peso total da segmentação, foi simulada uma rede de fluxo, onde foi enviado o máximo valor de fluxo possível entre o vértice *source* e o vértice *target*. Pelo teorema do fluxo máximo - corte mínimo, este valor é igual à capacidade que quando removida da rede conduz à situação onde mais nenhum fluxo passa entre a origem e o destino, ou seja, ao valor do corte mínimo.

Para representar o grafo foi usada uma lista ligada de adjacências e uma matriz de *V* linhas e 6 (majorante do grau dos vértices) colunas para as capacidades. Esta matriz tira vantagem da geometria do grafo para aceder à capacidade de cada aresta em tempo constante, *O(1)*.

Ao receber os inputs com os pesos de plano e de cenário de cada vértice, é logo enviado o máximo fluxo possível por todos os caminhos de tamanho dois (*source* → pixel → *target*), o que pode reduzir bastante o número de caminhos de aumento encontrados pelo algoritmo e consequentemente o seu número de iterações.

O algoritmo utilizado para determinar o fluxo máximo foi o de Edmonds-Karp, que é uma implementação do algoritmo de Ford-Fulkerson que garante que em cada iteração é usado o caminho de aumento mais curto, encontrado a partir de uma *BFS* (*breadth-first-search*), uma procura em largura com início no vértice inicial (*source*) e que visita todos os seus vizinhos, sendo que para cada vértice vizinho serão descobertos os seus vizinhos, e assim sucessivamente até ser encontrado o vértice final (*target*).

Para isto foram usados dois arrays auxiliares de tamanho do número de vértices, que guardam o *parent* e a quantidade de fluxo que consegue chegar a cada vértice, em cada *BFS*. Este fluxo é calculado determinando o mínimo entre a quantidade de fluxo que chega ao *parent* e a capacidade da aresta que o liga ao pixel atual.

Quando já não existirem caminhos de aumento, ou seja, o fluxo for máximo, é feita uma BFS com origem na *source* para determinar o corte mínimo e consequentemente classificar cada pixel. Um pixel será de cenário se foi visitado e será de primeiro plano caso contrário.

Análise Teórica

Para poder ser estudada a complexidade do algoritmo desenvolvido, é necessária a análise da complexidade das partes que o constituem.

Como descrito na solução, foram utilizados os algoritmos de procura *BFS* e Edmonds-Karp, caracterizados pelas complexidades *O(V + E)* e $O(V * E^2)$ respetivamente, sendo que *V* corresponde ao número de vértices e *E* ao número de arestas do grafo.

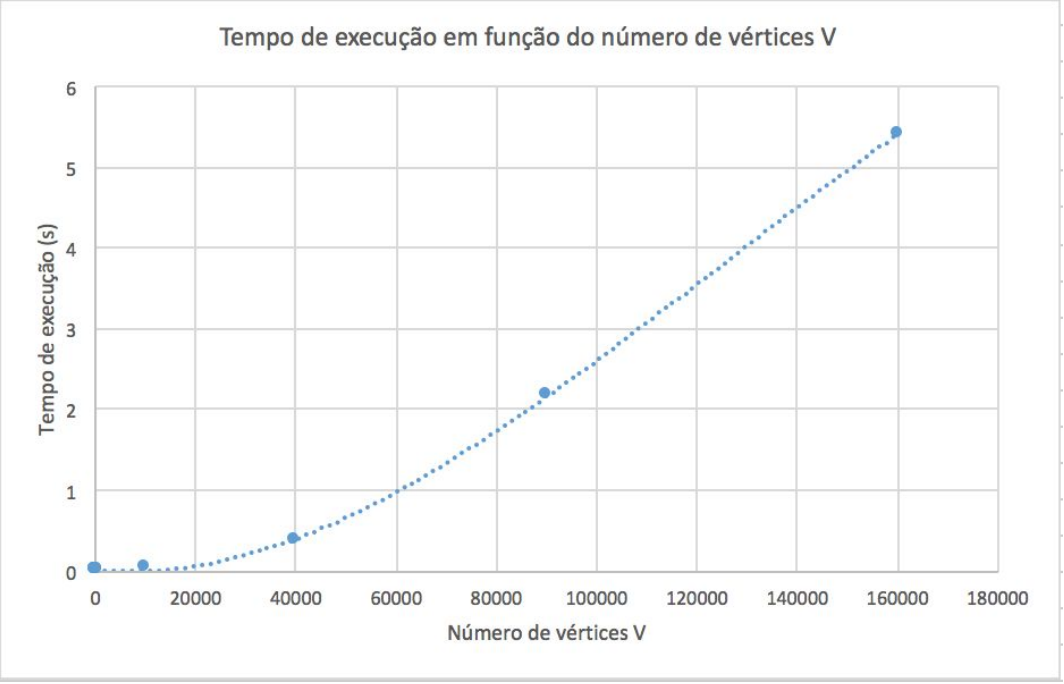
No algoritmo de Edmonds-Karp, como os grafos são densos, ou seja, $E > V$, cada iteração da BFS é executada em $O(E)$ e são feitas no máximo $V*E$ *BFSs*, pelo que a complexidade do algoritmo proposto será $O(V * E^2)$. Visto o número de arestas E ser no máximo $6V$ (ligação à *source*, ao *target*, e aos quatro vértices seus vizinhos - em cima, em baixo, na esquerda e na direita), a complexidade do algoritmo pode ser expressa como $O(V^3)$.

Durante o algoritmo, visto que apenas são utilizados *arrays* de tamanho V e que a informação acerca das arestas está guardada em estruturas de tamanho $E < 6V$, podemos concluir que a complexidade espacial da solução é $O(V)$.

Avaliação Experimental de Resultados

Para analisar o programa, foram realizados vários testes recorrendo aos testes facultados na pasta bigIO. Foi-se aumentando progressivamente o número de pixeis da imagem de modo a estudar o crescimento do tempo de execução do programa, tal como fornecido nos testes. Foi utilizado o comando *time*, de forma a extrair o *real time* quatro vezes, tendo sido calculada a média aritmética entre estes para obter os valores apresentados no gráfico.

Número de vértices (V)	Tempo de execução (s)
20	0,009
72	0,01
156	0,01
272	0,011
400	0,011
10000	0,05
40000	0,377
90000	2,168
160000	5,403



O tempo de execução obtido é aproximadamente $O(V^3)$, tal como previsto na análise teórica.

Comentários e Referências

Para o desenvolvimento da solução ao problema proposto, recorreu-se à linguagem C devido à sua eficiência e controlo sobre a memória utilizada e estruturas necessárias.

Os algoritmos BFS e Edmonds-Karp utilizados foram baseados no pseudocódigo e nas definições sugeridas nas aulas teóricas de ASA 16/17, no livro "*Introduction to Algorithms, Third Edition"* e nos sites :

https://en.wikibooks.org/wiki/Algorithm_Implementation/Graphs/Maximum_flow/Edmonds-Karp

https://sahebg.github.io/computersceince/maximum-flow-edmond-karp-algorithm-c-program-example/