Documentação TP1

Giovanna Louzi Belonia
Algoritmos I

17 de abril de 2019

1 Descrição do Problema

Busca-se criar um algoritmo que permite a alocação de alunos candidatos à universidades dado uma lista de preferência. O objetivo é satisfazer ambos os grupos, considerando que cada universidade tem uma nota mínima e que dão preferência à alunos com notas maiores; e que os candidatos tem uma lista de preferência de universidades, devendo ser alocados da melhor forma possível.
Os candidatos podem apresentar uma quantidade de preferências menor do que a quantidade de universidades existente e só serão alocados nas quais apresentou preferência e nas quais possui nota mínima para entrar.
As universidades podem ter mais alunos interessados do que quantidade de vagas, nesse caso, os v primeiros alunos com maiores notas (v sendo quantidade de vagas) serão escolhidos.

2 Estruturas de dados e algoritmos utilizados

2.1 TADs Utilizados

2.1.1 Preferências

Foi criado uma lista de preferências (**lista_pref_t**) que será utilizada tanto por universidades quanto por candidatos. Essa lista tem como atributos a quantidade de elementos que possue (**num_elementos**) e um ponteiro para o primeiro e último elementos, sendo esses as preferências (**pref_t**) que possuem ponteiro para o elemento anterior e próximo e como atributos o **pref_id** e o **pref_pos**, cujo uso será explicado mais a frente.

As funções atribuídas à lista são:

- **cria_lista_pref:** inicializa a lista com todos seus atributos nulos;
- **insere_pref:** insere uma pref_t em uma lista de acordo com os dados passados como parâmetros;
- **remove_inicio:** remove o primeiro elemento da lista;
- **remove_fim:** remove o último elemento da lista retornando o pref_id do elemento removido;

- **prin_lista_pref:** printa os pref_id de todos os elementos da lista em ordem (utilizado para testes de funcionamento);
- **destroi_lista_pref:** desaloca a memoria utilizada pela lista.

2.1.2 Universidades

Cada universidade foi representada pela estutura **univ_t** que tem como atributos **vagas**, **nota_min** e uma lista de preferências **lista_cand** que guardará os candidatos que estão sendo alocados à essa universidade.
O conjunto de universidades foi guardado em um vetor **vet_univ** de tamanho **u** (quantidade de universidades) em que a universidade **i** é guardada na posição **i-1** do vetor.

2.1.3 Candidatos

Cada candidato foi representado pela estrutura **cand_t** que tem como atributos **nota**; **aloc** (que indica o indíce de qual universidade foi alocado ou se não foi alocado - valor de aloc = -1); e uma lista de preferências, **lista_univ**, que guarda, na ordem, qual a preferência dos candidatos por qual universidade.
O conjunto de candidatos foi guardado em um vetor **vet_cand** de tamanho **c** (quantidade de candidatos) em que o candidato **i** é guardado na posição **i-1** do vetor.

2.2 Algoritmos utilizados

2.2.1 Main

- **Inicialização do vetor universidade:** é lido do arquivo a quantidade **u** de universidades e inicializado um vetor **univ_t** desse tamanho. Para cada uma são atribuídas a quantidade de vagas e a nota mínima lidas do arquivo, e é inicializado a lista de candidatos (inicialmente nula);
- **Inicialização do vetor candidatos:** é lido do arquivo a quantidade **c** de candidatos e inicializado um vetor **cand_t** desse tamanho. Para cada um são atribuídos à nota o valor lido do arquivo e -1 ao aloc (não foi alocado ainda). Além disso, a lista de universidades é inicializada baseada nos índices lidos no arquivo (subtraindo 1 para adaptar a lógica de vetores) e a posição que tal universidade tem na lista (a primeira universidade de interesse é 0, a segunda 1, etc.);
- **Alocação dos candidatos:** é chamado a função **aloca()** que passa como paramêtro os vetores **vet_univ** e **vet_cand** e a quantidade **c** de candidatos. O funcionamento de tal função será explicada mais a frente;
- **Print do resultado:**
  - é impresso inicialmente os "Grupos com alocacao". Para isso, o vetor de candidatos é percorrido e sempre que seu aloc for diferente de -1 é impresso a posição no vetor + 1 (que equivale ao índice do candidato) e o valor de aloc + 1, que equivale ao índice da universidade em que foi alocado;
  - é impresso depois os "Candidatos nao alocados" que são todos os candidatos do vetor candidatos com aloc = -1. É impresso então a posição no vetor + 1, que indica o índice desses candidatos.

- **Liberação da memória:** é chamado a função **destroi_lista_pref()** pra todas as universidades e candidatos de forma a desalocar a memória gasta em suas respectivas listas de preferência.

2.2.2 Alocação

A função para alocar tentará colocar cada candidato **i** do vetor **vet_cand** em sua universidade de preferência. Se o candidato não conseguir ser alocado na primeira opção (seja por não ter nota maior que a mínima ou por ter outras pessoas com uma classificação melhor e a universidade já cheia) tentaremos alocar o candidato na próxima universidade de preferência, seguindo assim até que consiga ser alocado ou que a lista de preferências dele acabe.
Se um aluno conseguir ser alocado em uma universidade já cheia, significa que, ou tem uma nota maior que a menor nota já alocada, ou então as notas eram iguais mas a universidade tem uma preferência maior na lista do candidato a ser alocado (por exemplo era a segunda opção dele enquanto do outro era a terceira). Nesses casos, o candidato que já estava alocado deixará de estar, e devemos, então, voltar ao primeiro passo e tentar alocá-lo nas próximas preferências de universidade da sua lista.

2.3 Makefile

Para gerar o compilável tp1 é só utilizar o comando *make compile*.

3 Análise de complexidade

3.1 Espaço

Será gasto um espaço **u** para o vetor de universidades e **c** para o vetor de candidatos. Cada lista de preferência de cada candidato pode ter um tamanho máximo de **u**, logo um espaço de **u*c**. Já a lista de preferências de universidades somadas dão no máximo **c**, já que cada candidato só pode ser alocado uma vez no total. Logo, o espaço total gasto é $O(c + u + u * c + c) = O(u * c)$.

3.2 Tempo

Para preencher os vetores de universidades e de candidatos serão gastos respectivamente $O(u)$ e $O(c)$. A função de alocar gastará para cada **c** candidato no pior caso, $O(u)$, pois terá que analisar todas as preferências, que pode ter um valor máximo de **u**, logo, terá um gasto de $O(u * c)$.
Obs: a inserção em ordem na lista de preferências se assemelha ao Insertion Sort, que, no pior e médio caso tem $O(n^2)$. Entretanto, como a lista está sempre ordenada ao inserir um novo elemento é $O(n)$, por isso podemos desconsiderá-lo na avaliação da complexidade já que $O(uxc)$ será maior.

4 Avaliação Experimental

Foram feitos 4 testes diferentes variando alguns pontos diferentes para fazer uma mínima análise experimental de como a mudança nos parâmetros de candidatos e univer-

sidade alteram na satisfação dos candidatos e na média das alocações.

4.1 Teste 1

Para o teste 1 foi criado o arquivo de universidades e candidatos de forma aleatória e alterado a quantidade de preferências que os candidatos poderiam ter até que atingissem a quantidade máxima possível (equivalente ao número total de universidades).

Tabela 1: Análise da satisfação e alocação dos candidatos a medida que aumentamos a quantidade de preferências possível.

Quant. de Preferências	Grau Satisfação	Taxa Alocação
2	0,53	0,36
3	0,63	0,44
4	0,63	0,44
5	0,63	0,44
6	0,66	0,48
7	0,66	0,48
8	0,66	0,48

Com esse teste podemos perceber que a taxa de alocação e o grau de satisfação aumentaram, porém de forma lenta e com uma mudança pequena.
Isso ocorreu porque os candidatos que não haviam sido alocados até então, ao colocarem mais opções, terão maior chance de conseguirem uma vaga em outras universidades. Dessa forma, esses que foram alocados depois terão, agora, satisfação > 0. Porém, esse aumento trará uma diferença pequena na média de satisfações já que são alocados em universidades de menor preferência, enquanto os que já haviam sido alocados mantém a satisfação que possuiam anteriormente.
A taxa de alocação muda apenas quando candidatos antes não alocados conseguem ir para uma universidade. O padrão de como isso pode mudar (nesse caso ao trocar de 2 para 3 e de 5 para 6 preferências) depende muito das entradas tanto de candidatos quanto de alunos.

4.2 Teste 2

Para o teste 2 foi criado um arquivo aleatório de candidatos, com quantidade de preferências variado, e um arquivo de universidades semi-aleatório, já que inicialmente foram definidas apenas uma vaga para todas as universidades e depois mudou-se aos poucos a média das vagas (com pequena variância).

Tabela 2: Análise da satisfação e alocação dos candidatos a medida que aumentamos a quantidade de vagas nas universidades.

Média de vagas	Grau Satisfação	Taxa Alocação
1	0,43	0,62
2,75	0,75	0,42
4,625	0,80	0,27

Com esse teste podemos perceber que o grau de satisfação aumentou com o aumento das vagas enquanto a taxa de alocação se reduziu. A drástica mudança no grau de satisfação se dá na possibilidade dos candidatos serem alocados nas universidades que tem maior preferência, já que o único impedimento passará a ser nota mínima, sem mais a concorrência de nota com outros candidatos que prefiram a mesma universidade.
A taxa de alocação diminuiu já que o número de vagas aumentou enquanto o de candidatos permaneceu o mesmo.

4.3 Teste 3

Para o teste 3 foi utilizado o mesmo txt de candidatos do teste 2, e, para o primeiro caso, o mesmo txt de universidades com a média de vagas de 4,625 do teste 2 (em que todos os candidatos foram alocados). Para os outros casos, foi-se aumentando as notas mínimas, colocando um limite inferior cada vez maior.

Tabela 3: Análise da satisfação e alocação dos candidatos a medida que aumentamos a nota mínima das universidades.

Médias nota mínima	Menor nota mínima	Grau Satisfação	Taxa Alocação
58,375	10	0,80	0,27
64,625	50	0,65	0,19
68,375	60	0,50	0,11
73,75	70	0,50	0,11
81,25	80	0,40	0,09
90	90	0,20	0,04

Com esse teste podemos perceber que o grau de satisfação e a taxa de alocação foram diminuindo gradativamente. Isso era o esperado já que, com o aumento das notas de corte, a chance de os candidatos terem uma nota menor do que a que a universidade pede, aumenta. Ou seja, ou eles não conseguirão se alocar em nenhuma ou não conseguirão alocar naquela de preferência, reduzindo o valor dos dois parâmetros analisados.

4.4 Teste 4

Para o teste 4 foi utilizado o mesmo txt de candidatos dos testes 2 e 3. Já para o primeiro txt de universidades foi escolhido as mesmas notas mínimas do primeiro caso do teste 3, porém a média de vagas por universidade foi menor (1,625). Para os outros casos, foi-se reduzindo as notas mínimas, colocando um limite superior cada vez menor.

Tabela 4: Análise da satisfação e alocação dos candidatos a medida que diminuímos a nota mínima das universidades.

Média notas mínima	Maior nota mínima	Grau Satisfação	Taxa Alocação
58,375	90	0,63	0,52
57,125	80	0,63	0,52
54,625	70	0,63	0,52
50	60	0,63	0,52
43,75	50	0,68	0,54
36,35	40	0,78	0,67
27,5	30	0,88	0,79

Com esse teste podemos perceber que o grau de satisfação aumentou um pouco e que a taxa de alocação também. Os candidatos que não haviam conseguido se alocar passam a conseguir por terem nota suficiente.
Obs: No caso em que a maior nota mínima é 50, os dois candidatos que não estavam alocados continuaram sem ir para uma universidade, entretanto, o grau de satisfação aumentou. Isso aconteceu, pois, com a redução da nota mínima, outro candidato já alocado pôde ir para uma universidade que prefiriria mais mas não conseguia entrar devido a sua nota.

4.5 Considerações finais

Os testes realizados anteriormente não servem como regra para todas as situações possíveis, já que muitas variáveis podem alterar o cálculo final do grau de satisfação e taxa de alocação. Para encontrar padrões de uma forma mais precisa, deveriam ter sido feitos muitos mais testes, modificando as muitas variáveis como a quantidade de universidades, de candidatos, média de notas, média de vagas, e assim sucessivamente; além de todas as cobinações possíveis dentre essas mudanças.