MC558A - Projeto e Análise de Algoritmos II

Lehilton Pedrosa
Murilo de Lima (PED C)

Exercício de Programação IV

- **Prazo de submissão:** 21 de maio às 23:59:59
- O exercício deve ser implementado em C ou C++
- Número máximo de submissões: 10
- Tempo máximo de execução: 10s

Bar da Inês

Depois de fazer muito sucesso como cantora, dançarina, filósofa e teóloga poliglota, Inês decidiu juntar suas economias para abrir um bar. O bar conta com uma excelente infraestrutura, com pista de dança, luzes de neon e LED e equipamento de som. No entanto, devido à sua agenda muito requisitada, Inês não terá como gerenciá-lo a todo o tempo. Ela decidiu, então, alugar seu espaço para eventos em alguns períodos do ano.

Qual foi a surpresa de Inês ao perceber que os pedidos de reserva não paravam de chegar! Em cada pedido de reserva, um cliente define em qual período deseja utilizar o espaço e propõe um valor de aluguel. Infelizmente o sistema que a Noob Corporation desenvolveu para gerenciar os pedidos não registra o horário em que um cliente solicitou uma reserva, então Inês não pode priorizar quem fez o pedido primeiro. Para o mês de junho, então, ela decidiu atender às reservas que lhe proporcionem o maior lucro possível. A única restrição é que ela não pode atender reservas que tenham conflito de horário.

Seu objetivo é ajudar Inês a encontrar um conjunto de reservas que não tenham conflito e que maximizem o lucro. Para resolver o problema, você deve **obrigatoriamente** utilizar como *caixa-preta* a implementação que disponibilizamos de um algoritmo que obtém um caminho mínimo em um grafo acíclico dirigido com pesos nas arestas. Mesmo que você saiba resolver o problema de outra forma mais eficiente, o objetivo deste laboratório é exercitar a redução entre problemas. **Implementações que utilizem outras estratégias receberão nota zero, independentemente da qualidade do trabalho.**

Links para download: versão em C / versão em C++. Não é permitido modificar o código do algoritmo fornecido, nem a definição das estruturas de dados. Você deve modificar o arquivo-exemplo lab4.c ou lab4.cpp para resolver o problema, e **submeter apenas esse arquivo** no SuSy. O SuSy irá linkar seu código com o algoritmo fornecido automaticamente.

Entrada: na primeira linha da entrada é dado um inteiro n, que indica o número de pedidos. Os pedidos são numerados de 1 a n. Você pode supor que **$1 \leq n \leq 5000$**.
A seguir são dadas n linhas; cada linha contém três números inteiros i, f e v, que indicam que foi feito um pedido para o período [i, f) com valor de aluguel proposto v. Você pode supor que $i < f$, que $1 \leq i, f \leq 10^9$, e que $0 \leq v \leq 10^5$.

Saída: uma linha com um único número inteiro correspondendo ao valor do lucro obtido, seguida de uma linha com os números dos pedidos que serão atendidos, em ordem crescente de tempo de início e separados por espaço. **Note que pode haver mais de uma solução com o mesmo valor; qualquer solução ótima estará correta.**

Exemplo:
Entrada:
3
1 3 20
2 5 50
4 7 40

Saída:
60
1 3
(A solução ótima consiste em agendar os pedidos 1 e 3. Note que o pedido 2 tem conflito com os pedidos 1 e 3.)

Relatório: você deve incluir, no cabeçalho do arquivo do seu código, um comentário com 100 a 300 palavras, explicando sua redução. Não é necessário fazer uma prova formal, mas você deve argumentar por que sua solução funciona.
Sugestão: escreva o relatório antes de implementar o código; isso vai te ajudar a pensar melhor no problema. Após concluir a implementação, revise o relatório para checar se algo precisa ser modificado.

Observações:

- O SuSy utiliza o GCC 4.4.7 20120313 (Red Hat 4.4.7-17). São utilizadas as seguintes flags para compilação:
  - C99: `-std=c99 -pedantic -Wall -lm`
  - ANSI C: `-ansi -pedantic -Wall -lm`
  - C++: `-ansi -pedantic -Wall -lm`
- Você deve implementar estruturas de dados eficientes, com consumo de memória $O(n^2)$.
- Os passos de transformação da sua redução devem executar em tempo $O(n^2)$. Lembre-se que alocação de memória (mesmo memória estática alocada na pilha) influi na complexidade de tempo.
- A nota do exercício é proporcional ao número de casos de teste que você acertar; são dados 10 casos de teste, sendo 7 abertos e 3 fechados, valendo 1 ponto cada.
- No entanto, **implementações com complexidade de memória ou tempo fora do especificado receberão nota zero no exercício**.
  Isto poderá ser verificado através de casos de teste fechados adicionais, com tamanho de entrada maior, executados fora do SuSy pelo monitor.
- Você pode utilizar as bibliotecas-padrão do C e as estruturas de dados da biblioteca-padrão do C++.
- **Trechos de código copiados da Internet ou dx coleguinha configuram plágio.**
- Sugerimos que você use scanf para fazer a leitura da entrada, a fim de garantir que seu código execute no tempo especificado.
- Seu código deve estar identado, modularizado e bem comentado. Identifique-se e deixe claro quais estruturas de dados e algoritmos foram utilizados.
