TP3 - A Cifra Carmesim

17 pontos

Entrega: 23/06/2024

1 Objetivo do trabalho

Neste trabalho, vamos exercitar tópico relativos ao último terço do curso: paradigmas de desenvolvimento de algoritmos.

Serão fornecidos alguns casos de teste para que você possa testar seu programa, mas eles não são exaustivos! Podem haver situações que não são ilustradas por eles; cabe a você pensar em novos casos e garantir que seu programa esteja correto e que ele implemente um algoritmo de complexidade adequada.

1.1 Informações importantes

O código fonte do seu trabalho deve estar contido em um **único** arquivo na linguagem C++ e deve ser submetido via Moodle na tarefa `Entrega TP3` até o dia 23/06/2024. Você terá ∞ tentativas para conseguir a nota total de execução; **apenas a última submissão será levada em conta para fins de avaliação**. Você não terá acesso a todos os casos de teste; determinar estratégias para testar seu programa e suas ideias faz parte do trabalho. Envios com atraso serão aceitos; leia a Seção 5 para a política de atrasos.

Plágio de qualquer natureza não será tolerado. Caso qualquer cola seja encontrada, seu trabalho será zerado e as demais providências cabíveis serão tomadas. Escreva seu próprio código, de maneira legível e com comentários apropriados; ele pode ser útil no futuro próximo.

2 Definição do problema

Os Reis Amarelos se tornou um sucesso gigantesco na Bacônia, ao ponto de que os jogadores clamam por mais conteúdo educativo sobre o glorioso reino das capivaras. Com isso em mente, a desenvolvedora (uma empresa conhecida apenas pela sigla ALG) de ORA optou pelo modelo de expansões, onde mais personagens, monstros, missões, e mecânicas são disponibilizados periodicamente em um grande pacote.

O último grande lançamento da ALG é a expansão *A Cifra Carmesim*, onde os jogadores devem desvendar os mistérios naturais, sociais, e arcanos da Bacônia de outrora. ALG levou tão a sério seu conceito que a própria caixa d'*A Cifra Carmesim* é um quebra-cabeça!

A caixa pode ser descrita como uma malha circular com L linhas e C colunas, formando uma malha com **até** $L \times C$ cristais, cada um no encontro de uma linha com uma coluna. Os cristais são a chave do quebra-cabeça e devemos ativá-los da maneira mais eficiente possível. Cada cristal tem uma cor, que indica o quão vermelho é o seu brilho quando ativado. Além disso, cada cristal tem algumas conexões, por exemplo o cristal na coordenada (i, j) é conectado há um subconjunto dos cristais $\{(i, j-1), (i-1, j), (i, j+1), (i+1, j)\}$; todas essas coordenadas são consideradas módulo a dimensão apropriada, ou seja, a borda esquerda pode se conectar à borda direita e a borda de baixo à borda de cima. Para resolver o mistério da caixa, temos de conseguir o brilho mais vermelho possível sem ativar nenhum par de cristais adjacentes. Resolva este problema para decifrar a *Cifra Carmesim*! Veja abaixo um exemplo de caixa com $L = 3$ e $C = 4$.

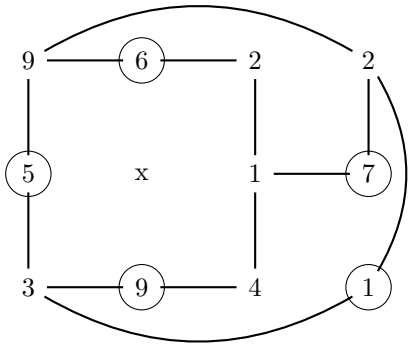

Figure 1: Uma possível caixa de entrada, com uma solução ótima correspondente aos cristais circulados. Cristais faltantes são marcados com um x.

3 Dicas

Um segundo equivale a, aproximadamente, 10^8 operações em um computador moderno. Ou seja, um algoritmo $\mathcal{O}(N)$ para $N \leq 10^6$ cabe em um segundo, mas um algoritmo $\mathcal{O}(N^2)$ não cabe.

Operações de entrada e saída são extremamente caras: use as funções `scanf` e `printf` ao invés das funções `cin, cout`.

Tempo e espaço são apenas dimensões do universo.

4 Casos de teste

4.1 Formatado da Entrada

Cada caso de teste é composto por várias linhas, que representam uma caixa da *Cifra Carmesim*. A primeira linha contém três inteiros L, C, e N, o número de linhas, o número de colunas, e o número de cristais na caixa, respectivamente. É garantido que $1 \leq L \leq 600$, $1 \leq C \leq 6$, e $1 \leq N \leq L \times C$.

Seguem-se então N linhas; a i-ésima dessas linhas contém 7 inteiros $x_i, y_i, v_i, d_i, c_i, e_i, b_i$ onde (x_i, y_i) corresponde à coordenada do i-ésimo cristal e v_i corresponde ao brilho do mesmo. O inteiro $d_i \in \{0, 1\}$ é igual a 1 se e somente se o cristal i se conecta ao cristal imediatamente à sua direita. O mesmo vale para c_i e o cristal acima, e_i e o cristal à esquerda, e b_i e o cristal abaixo. É garantido que cada cristal é descrito apenas uma vez e que $1 \leq v_i \leq 100$ para todo $i \leq N$.

4.2 Formato da Saída

A primeira linha da saída deve conter dois inteiros, S e V que representam, respectivamente, o número de cristais na solução e o valor total da mesma. Em seguida devem haver S linhas, que descrevem os cristais utilizados. Na i-ésima dessas linhas devem haver apenas dois inteiros, x_i, y_i, que correspondem às coordenadas do i-ésimo cristal utilizado na solução.

4.3 Limites de execução

Para qualquer caso de teste, seu código deve imprimir a resposta em, no máximo, 3 segundos. Seu programa deve usar menos de 100MB de memória. Estruturas de dados devem ser alocadas sob demanda; ou seja, não faça vetores estáticos gigantescos para grafos com poucos vértices. Todas as avaliações serão feitas automaticamente via VPL. Programas que não aderirem a essas restrições para um teste terão a nota do mesmo zerada.

4.4 Exemplos

4.4.1 Exemplo da Figura 1

Entrada	Saída
3 4 11	5 28
1 1 9 1 0 1 1	1 2
2 1 5 0 1 0 1	2 1
3 1 3 1 1 1 0	3 2
1 2 6 1 0 1 0	2 4
3 2 9 1 0 1 0	3 4
1 3 2 0 0 1 1	
2 3 1 1 1 0 1	
3 3 4 0 1 1 0	
1 4 2 1 1 0 1	
2 4 7 0 1 1 0	
3 4 1 1 0 0 1	

5 Atrasos

O trabalho pode ser entregue com atraso, mas será penalizado de acordo com a seguinte fórmula, onde d é o número de dias atrasados:

$$\Delta(d) = \frac{2^{d-1}}{0.32}\% \tag{1}$$

Por exemplo, com um atraso de quatro dias e uma nota de execução de 70% do total, sua nota final será penalizada em 25%, ficando assim igual a $70 \cdot (1 - \Delta(d)) = 52.5\%$. Note que a penalização é exponencial, e um atraso de 6 dias equivale a uma penalidade de 100%.

6 Errata

- Ajuste nos limites de L e C.