MAC0345 - Desafios 2
Editorial Lista 12

Enrique Junchaya

June 28, 2023

A. Zoning Restrictons.
Tópicos: fluxo

Se o problema não tivesse as restrições de altura máxima em intervalos, fazer com que todo mundo tenha a altura máxima h seria o melhor possível. Contudo, com essas restrições, nem sempre isso vale a pena.

Tentemos modelar, então, o valor de uma configuração de prédios em que o i-ésimo prédio tem altura a_i. Ignorando, temporariamente as restrições nas alturas máximas, teríamos que o valor dessa configuração é $\sum_{i=1}^{n} a_i^2$. Sabemos que adicionar os custos de ultrapassar as alturas máximas, faria com que tivéssemos que tirar esses custos da somatória. Mas tanto fluxo quanto mincut dão valores positivos. Não é possível tanto adicionar quanto tirar coisas. Só podemos ter coisas com o mesmo sinal. Nesse sentido, pensemos como modelar de tal forma que só precisemos minimizar o que vamos tirar.

Veja que podemos reescrever o valor da somatória dos prédios como $\sum_{i=1}^{n} h_i^2 - (h_i^2 - a_i^2) = nh_i^2 - \sum_{i=1}^{n}(h_i^2 - a_i^2)$. Suponhamos que C é o custo total por ter ultrapassado os limites de altura. O valor final da configuração de prédios seria $nh_i^2 - \sum_{i=1}^{n}(h_i^2 - a_i^2) - C$. Como nh^2 é constante, o problema vira minimizar $\sum_{i=1}^{n}(h_i^2 - a_i^2) + C$. Note que essa conta está composta apenas de parcelas positivas, então podemos tentar modelar o problema como corte mínimo.

Sejam X e Y as duas partições de um st-corte tal que $s \in X$ e $t \in Y$. Vamos modelar os prédios com $h + 1$ vértices por cada um deles. O vértice v_{ij} representa a altura j do prédio i. Desta forma, vamos modelar a rede tal que $v_{ij} \in X$ se e somente se o prédio i tem altura maior ou igual a j. É claro que se o vértice v_{ij} está em X então também o vértice v_{ij-1} tem que estar e em X. Podemos forçar isso adicionando arestas de capacidade infinita de v_{ij} a v_{ij-1}. Se o prédio i tem altura exatamente j, então $v_{ij} \in X$ mas $v_{ij+1} \notin X$. Assim, podemos modelar o valor da altura do prédio i colocando arestas de capacidade $h^2 - j^2$ do vértice v_{ij} ao prédio v_{ij+1}. Finalmente, para modelar os custos de uma restrição l_i, r_i, x_i, c_i, vamos criar um vértice u_i e uma aresta de capacidade c_i de u_i a t. Queremos forçar esse vértice estar em X se existe algum prédio $l_i \leq j \leq r_i$ com altura pelo menos x_i, ou seja, tal que $v_{jx_i} \in X$. Então, podemos colocar arestas de capacidade infinita de v_{jx_i} a u_i para todo $l_i \leq j \leq r_i$.

B. Down went the Titanic.
Tópicos: fluxo

É direto pensar numa modelagem com fluxo para resolver esse problema. Temos uma *source* s e um *sink* t. Ligamos s a todos os vértices representados por '.' com uma aresta de capacidade 1 para representar a única pessoa que começa nesse vértice. Ligamos os vértices representados por '`#`' a t com capacidade P, representando que no máximo P pessoas podem esperar a serem resgatados nesse vértice.

A dificuldade está em modelar o resto da rede. Veja que os vértices '~' e '`@`' são como arestas que só deixar passar 1 e infinito de fluxo respectivamente. Porém, não é tão simples modelar isso. Colocar todas as arestas saindo e entrando desses vértices com tal capacidade não é suficiente porque queremos o fluxo passa pelas arestas e não tem restrição no vértices. Veja, pode ter um caminho entrando de cima e indo pra baixo e outro entrando da esquerda e indo pra direita num vértice '~' e isso não quebraria as capacidades

das arestas apesar de ter mais de um caminho passando pelo vértice. Devemos tentar colocar restrições nos vértices.

A ideia é modelar esses vértices como se fossem arestas. Para tal, duplicaremos eles: uma cópia com as arestas que entram no vértice e outra com as que saem. Depois, adicionaremos uma aresta da primeira cópia para a segunda cópia com a capacidade do vértice. Não é difícil verificar que essa modelagem funciona para adicionar capacidades nos vértices.

C. Fast maximum flow.
Tópicos: fluxo

Só achar o fluxo máximo de 1 até N. Note que implementações não muito eficientes de fluxo não conseguem passar a questão. Em geral, as questões de fluxo podem ser resolvidas usando um algoritmo eficiente como caixa preta. Assim, não se sintam culpadas de só copiar e colar um *template* de, por exemplo, Dinic para passar essa ou outras questões de fluxo. O mais relevante é saber modelar a rede de fluxo mesmo.

D. Coconuts.
Tópicos: fluxo

Vou chamar condição *coconut* à condição da historinha do problema que vai ser votada como verdadeira ou falsa. Vou chamar de T pela verdade da condição *coconut* e F o voto pela falsidade dessa condição.

Vamos modelar esse problema com *min-cut*. Lembre que o problema de corte mínimo é calcular o menor *st*-corte (s é o *source* e t é o *sink* da rede). Em outras palavras, queremos dividir o grafo em duas partes, X e Y, tal que $s \in X$ e $t \in Y$ e o custo das arestas que saem de X e entram em Y.

Seja (X, Y) uma partição de um *st*-corte. Queremos que todos os que votarão T estejam em X e os que votarão F estejam em Y. Desta forma, se adicionamos arestas (de capacidade 1) entre os amigos, um *st*-corte vai ter o custo dos amigos que votarão distinto (note que uma aresta de amizade aparece nas duas direções no corte, mas só contamos as arestas que SAEM do corte no custo deste). Faltaria então modelar o custo de votar contrário as suas crenças. Para isso adicionarmos uma aresta de capacidade 1 $s \to v$ para todo vértice v que representa a alguém que acredita na condição *coconut* e arestas de capacidade 1 $t \to u$ para todo vértice u que não acredita nessa condição. Não é difícil ver que isso incrementa $+1$ ao custo do corte por cada pessoa que vota contrário às suas crenças.

E. Petya and Graph.
Tópicos: fluxo

Podemos reduzir esse problema ao problema de Projetos e Instrumentos visto em aula: os projetos são as arestas e os instrumentos para fazer uma aresta são os dois vértices incidentes nela.

Se você quiser relembrar o problema de Projetos e Instrumentos, veja o editorial oficial.

F. Soldier and Traveling.
Tópicos: fluxo

Note que se não tivesse a restrição de que cada soldado pode se movimentar usando no máximo uma aresta, o problema seria simplesmente verificar se $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$. Como temos essa restrição, essa condição não é suficiente, mas é importante considerar que é necessária.

Como temos uma configuração inicial e queremos chegar numa configuração final, podemos pensar num grafo bipartido (X, Y), $|X| = |Y| = n$, em que os vértices em X representam o estado inicial do grafo e os vértices de Y o estado final do grafo. Assim, se modelamos o fluxo dos soldados com uma rede de fluxo, poderíamos adicionar, para toda aresta $uv \in G$ (G é o grafo original), uma aresta direcionada $u_1 \to v_2$ no grafo bipartido onde u_1 representa o estado inicial de u e v_2 o estado final de v. A capacidade dessa aresta tem que ser grande o suficiente para deixar a todos os soldados usar ela pois não temos restrições nas arestas (pode colocar infinito de capacidade sem problema). É importante também considerar que um

soldado pode permanecer na mesma posição. Para isso, adicione uma aresta $u_1 \to u_2$ para todo vértice u de G.

Finalmente, devemos modelar quantos soldados deve ter cada vértice na configuração inicial e quantas tem que ter na configuração final. Pode fazer isso colocando arestas $s \to i_1$ com capacidade a_i e arestas $i_2 \to t$ com capacidade b_i. Não é difícil ver que o problema tem solução se e somente se o fluxo máximo de s a t é igual a $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$. Para saber quantas pessoas foram de um vértice i a outro j basta ver o fluxo passando pela aresta $i_1 j_2$.