***INDICE***

|  |  |
| --- | --- |
| **Algoritmos gananciosos**   * Problema do troco * Escalonamento de atividades | **Grafos – introdução**   * Dirigidos / não dirigidos * Simples / pesados / bipartidos * Densidade * Caixeiro viajante - vértices * Representação grafos |
| **Divisão e Conquista**   * Merge sort * Quick sort * Cálculo de Xn * Pesquisa binária | **Grafos – pesquisa e ordenação**   * DFS – profundidade O(V + E) * BFS – largura O(V + E) * Ordenação topológica O(V + E) |
| **Algoritmos de retrocesso**   * Soma de subconjuntos   Poda de pesquisa | **Caminho mais curto – 1**   * Grafo dirigido não pesado * Grafo dirigido pesado – Dijkstra   O((V + E)log(V))   * Decrease key – eficiência * Bellman-Ford (pesos negativos) * Grafos acíclicos |
| **Programação dinâmica**   * Combinações * Problema da mochila * Fibonacci * Memoization * Sequências * Problema do troco com stock limitado | **Caminho mais curto – 2**   * Dijkstra bidirectional * Pesquisa orientada – A\* * Redes hierárquicas * Nós de trânsito * Floyd-Warshal (caminho mais curto entre todos os pares de vértices) |
| **Análise de algoritmos**   * Correção parcial e total * Pré-condições e pós-condições * Invariantes e variantes * Insertion sort O(n2) S(n) * Binary seach O(log(n)) | **Caminho mais curto – Gestão projetos**   * Menor tempo de conclusão * Último tempo de conclusão * Folga nas atividades |
|  |  |
| **Minimum spanning tree**   * Algoritmo Prim   O(V2) - s/ lista prioridade  O(ElogV) – c/ lista prioridade   * Algoritmo Kruskal O(ElogE)   **Grafos - conectividade**   * Grafos não dirigidos   + DFS   + Pontos de articulação   + Low() * Grafos dirigidos   + Componentes fortemente conexas   + Árvore de expansão | **Circuitos de Euler & problema do carteiro chinês**  Carteiro chinês para grafos não dirigidos e dirigidos - arestas  **Algoritmos de pesquisa - strings**   * Naïve O(TP) * Knuth-Morris-Pratt O(T+P) * Pesquisa aproximada – distância entre duas strings O(PT) |
| **Grafos – fluxo máximo e redes de transporte**   * Redes com múltiplas fontes e poços * Ford-Fulkerson O(FE)   + Edmonds-Karp O(VE2) * Dualidade entre fluxo máximo e corte mínimo | **Algoritmos de pesquisa – compressão**   * Huffman O(nlog(n))   **NP Completos – introdução**   * NP difícil & NP complete |
| **Grafos – fluxo mínimo e redes de transporte**   * Melhoramento – conversão do grafo de resíduos * Aplicação a problemas de emparelhamento   **Problema de emparelhamento e circuitos estáveis**   * Gale Shapley O(n2) | **NP Completos – Redução de problemas**   * Redes com múltiplas fontes e poços * Redução de UHC – DHC * Vertex – Cover * Independent Set * Problema da caminhada |

**NPs**

**Hamiltonian Cycle**

Um caminho hamiltoniano é um caminho que permite passar por todos os vértices de um grafo G, não repetindo nenhum, ou seja, passar por todos uma e uma só vez por cada. Caso esse caminho seja possível descrever um ciclo (partida e chegada no mesmo ponto), este é denominado ciclo hamiltoniano.

O problema de decidir se um dado grafo hamiltoniano é completo em NP, o que significa que é pouco provável que exista um algoritmo polinomial para o problema.

**Subset Sum Problem**

Dado um conjunto de inteiros positivos, S, há um subconjunto, S’ em S, tal que a soma dos elementos de S’ seja k?

**Clique Problem**

Um clique de um grafo não dirigido é um subconjunto dos seus vértices, tal que, para quaisquer pares de vértices u e v neste subconjunto, existe uma aresta do grafo que liga os vértices u e v (grafo completo). O problema de otimização consiste em encontrar um clique de tamanho máximo.

Dado um grafo não dirigido G=(V,E) e um k∈ℕ, verificar se G tem um clique de tamanho ≥ k?

**Vertex Cover Problem**

Uma cobertura de vértices de um grafo G = (V, E) é um subconjunto VC ⊆ V, tal que toda aresta (a, b)∈ E é incidente em pelo menos um vértice u ∈ VC .

Vértices em VC “cobrem” todas as arestas em G.

O grafo G tem uma cobertura de vértices de tamanho ≤k?

**Independent Set**

Um conjunto independente de um grafo G = (V, E) é um subconjunto VI ⊆ V, tal que não há dois vértices em VI que partilham uma aresta de E

u, v ∈ VI não podem ser vizinhos em G

O grafo G tem um conjunto independente de tamanho ≥k?

**Problema da Caminhada**

Considere um grafo não dirigido G, admitindo arestas paralelas e anéis, com pesos inteiros positivos nas arestas, no qual se distingue um vértice home.

O problema da caminhada (Jogging (J)) consiste em verificar se existe um caminho de peso total k, começando e terminando em home, sem repetir arestas.

Prove que J é um problema NPC, sabendo-se que o problema da soma de subconjuntos é NPC.

**Problema da partilha de viaturas**

Um grupo de pessoas pretende organizar os transportes em viatura própria (ida e regresso) para uma atividade de lazer num ponto definido, minimizando o consumo de combustível.

• Assumir que o consumo depende apenas da distância percorrida ν

* Para esse efeito, pessoas partindo de casas diferentes nas suas viaturas podem encontrar-se em pontos intermédios, deixando aí um dos carros (procedendo de forma inversa no regresso).

• Assumir que é possível deixar o carro em qualquer ponto ν Mostrar que é um problema NP-completo, sabendo-se que o problema da Árvore de Steiner em Grafos (ver slide seguinte) é NP-completo

**Problema da arvore de Steiner**

Seja G = (V, E) um grafo não dirigido com pesos não negativos ν Seja S ⊆ V um subconjunto de vértices, chamados terminais. ν Uma árvore de Steiner é uma árvore em G que contém todos os vértices de S. ν Problema de otimização: encontrar uma árvore de Steiner de peso mínimo

• É o mesmo que uma árvore de expansão mínima, no caso de S = V ν Problema de decisão (com pesos inteiros): determinar se existe uma árvore de Steiner de peso total que não excede um número natural k pré-definido

• Sabe-se que é um problema NP-completo

Problema de decisão: é possível efetuar o transporte com distância total percorrida ≤ k, pelo conjunto de viaturas?

**Resposta base**

*Para provar que é NP COMPLETE:*

A classe de problemas NP é definida por todos os problemas que podem ser verificado por um algoritmo de tempo polinomial.

Um problema de decisão A é NP-completo se (i) A € NP; e (ii) qualquer problema A’ € NP é redutível em tempo polinomial a A (A’ ≤p A).

Prova:

1. Provar que o problema está na classe NP
   1. É de decisão – problema cujo output ou resposta deve ser um simples “SIM” ou “NÃO”
   2. Uma solução é verificável em tempo polinomial
      1. [ Aplicar ao caso em concreto ] Ex:  
         Verificar em tempo linear ou constante se o nr de alunos na solução é >= a k  
         Percorrer todas as combinações de alunos na solução dois a dois e verificar se para cada combinação esses dois alunos ainda não participaram numa atividade em conjunto. Verificar um par pode ser feito em tempo constante com array 2D.
2. Provar que é NP Hard:

*Para provar que é NP HARD:*

Um problema NP-Hard é um problema que satisfaz a propriedade (ii) mas não necessariamente a propriedade (i).

* + Para provar que é NP-HARD tenta-se reduzir A’ problem, que já é provado ser um problema NP-completo (NP-hard portanto), ao nosso problema.
  + Descobrir transformação R que converte o input X do problema proposto ao input Y do problema A’, Y = f(X), em tempo polinomial.

Problema A -> [Aplicar ao caso em concreto A] Ex: formar turma

Input X -> lista de crianças.

Participou(A,B) -> Função booleana que retorna verdadeiro se a criança A já esteve com a criança B na mesma atividade de grupo; falso em caso contrário.

Tamanho k da turma que se pretende para limite inferior.

Problema A’ -> [Aplicar ao caso em concreto A’] Ex: independente set

Input Y -> Grafo G = (V,E)

Tamanho n do conjunto independente que se pretende para limite inferior.

Transformação R tal que Y = R(X) e que é feito em tempo polinomial (conversão de entradas e saídas):

* Cada vértice em G será uma criança e cada aresta entre duas crianças corresponderá à existência de uma atividade em grupo em que os dois tenham participado em conjunto.
* O conjunto independente terá apenas vértices tal que nenhum par de vértices no conjunto são adjacentes, ou seja, terá crianças que nunca participaram entre si em atividades de grupo.

Conclusão:

Assim, fica provado que a formação de turmas é um problema NP-Completo ou seja não resolúvel em tempo polinomial.