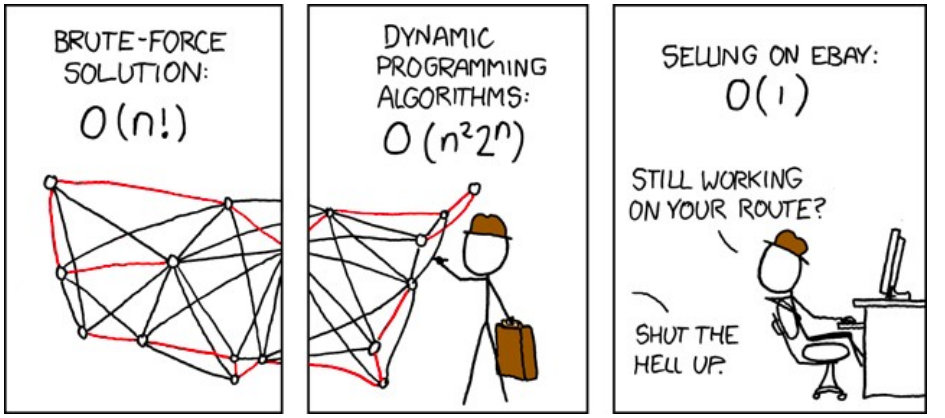
025089 – Projeto e Análise de Algoritmos

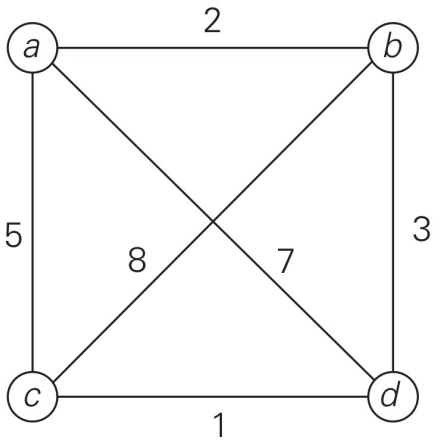
Aula 07

lizier@dc.ufscar.br

Exemplo 1

- Problema do caixeiro-viajante
  - Encontrar o menor circuito que passe por todas as cidades apenas uma vez

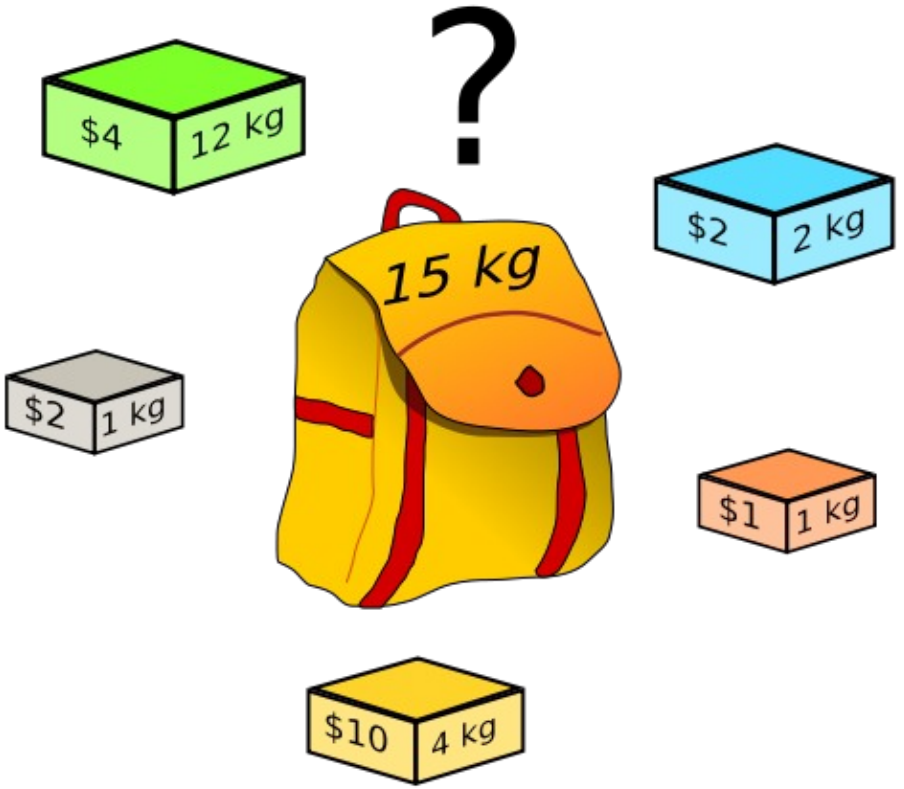




Tour	Length	
a ---> *b* ---> *c* ---> *d* ---> *a*	*l* = 2 + 8 + 1 + 7 = 18	
a ---> *b* ---> *d* ---> *c* ---> *a*	*l* = 2 + 3 + 1 + 5 = 11	optimal
a ---> *c* ---> *b* ---> *d* ---> *a*	*l* = 5 + 8 + 3 + 7 = 23	
a ---> *c* ---> *d* ---> *b* ---> *a*	*l* = 5 + 1 + 3 + 2 = 11	optimal
a ---> *d* ---> *b* ---> *c* ---> *a*	*l* = 7 + 3 + 8 + 5 = 23	
a ---> *d* ---> *c* ---> *b* ---> *a*	*l* = 7 + 1 + 8 + 2 = 18	

Exemplo 2

- Problema da mochila:
  - Dado um conjunto de itens, cada um com seu peso ***w*** e valor ***v*** definidos, encontrar o(os) subconjunto(s) de maior valor que cabe em uma mochila de tamanho ***K***



Exemplo 3

- Problema da atribuição:
  - Considere ***n*** pessoas e o custo de atribuir cada pessoa em um dos ***n*** empregos disponíveis. Determine qual o menor custo total se atribuirmos uma pessoa para cada emprego diferente

	Job 1	**Job 2**	**Job 3**	**Job 4**
Person 1	9	2	7	8
Person 2	6	4	3	7
Person 3	5	8	1	8
Person 4	7	6	9	4

Exemplo 1

- Algoritmo (força-bruta):
  - Gerar todas as possibilidades (quantas?)
  - Calcular o custo de cada uma
  - Escolher o menor caminho
- Representação
  - Grafo
  - Circuito

Exemplo 2

- Algoritmo (força-bruta):
  - Gerar todas as possibilidades (quantas?)
    - Subconjuntos que não cabem na mochila não são possibilidades
  - Calcular o custo de cada uma
  - Escolher o subconjunto de menor custo

- Backtracking
  - Árvore binária, inclui ou não o item na mochila

Exemplo 3

- Algoritmo (força-bruta):
  - Gerar todas as possibilidades de atribuição (quantas?)
  - Calcular o custo de cada uma
  - Escolher a atribuição de menor custo
- Representação:
  - Pessoa x Emprego

Exemplo 1

```
// int N, int g[][]
// int used[], int sol[], int atu[], int cmin

void copySolution() {
    for( int i=0; i<=N; i++)
        sol[i] = atu[i];
}
void tsp( int k, int c ) {
    if (k == N-1) {
        if( c + g[atu[k]][0] < cmin ) {
            cmin = c + g[atu[k]][0];
            copySolution();
        }
    } else {
        for( int i=0; i<N; i++)
            if ( !used[i] ) {
                used[i] = true;
                atu[k + 1] = i;
                tsp(k + 1, c + g[atu[k]][i]);
                used[i] = false;
            }
    }
}
```

Exemplo 3

```
// int int N, int pj[][]
// int cmin, int used[], int atu[], int sol[]
void copySolution() {
    for( int i=0; i<N; i++)
        sol[i] = atu[i];
}
void ap( int k, int c ) {
    if (k == N) {
        if( c < cmin ) {
            cmin = c;
            copySolution();
        }
    } else {
        for( int i=0; i<N; i++)
            if ( !used[i] ) {
                used[i] = true;
                atu[k] = i;
                ap(k + 1, c + pj[k][i]);
                used[i] = false;
            }
    }
}
```