α-Aproximação

- Um algoritmo H é uma *α-aproximação* para um problema P se
  - P é um problema de **minimização** e $\frac{z^H(I)}{z^*(I)} \leq \alpha \ \ \forall\, I$, *ou*
  - P é um problema de **maximização** e $\frac{z^*(I)}{z^H(I)} \leq \alpha \ \ \forall\, I$.

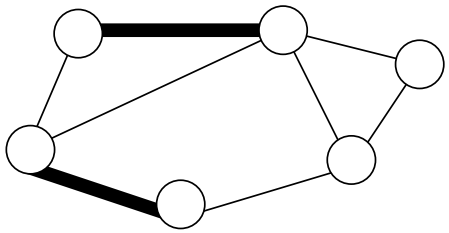
  Observação: α é sempre maior ou igual a 1.
- Um algoritmo H é uma *β-aproximação relativa* para um problema P se $\left|\frac{z^*(I)-z^H(I)}{z^*(I)}\right| \leq \beta$, para todo I.

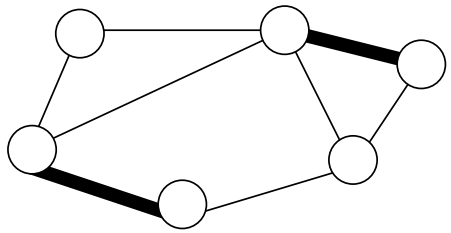
  Observação: P tem uma uma α-aproximação *sss* P tem uma β aproximação relativa.

$$\text{Min: } \beta = \alpha - 1 \qquad\qquad \text{Max: } \beta = \frac{\alpha - 1}{\alpha}$$

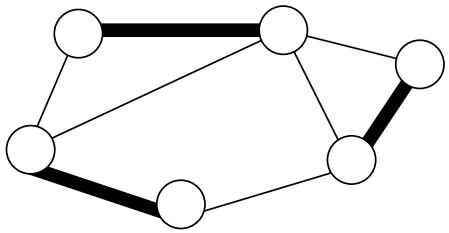
α-Aproximação

- ▷ Exemplo 1: Cobertura mínima de vértices (CV).
- ▷ *Definições*: emparelhamento em grafos.





MAXIMAL



MAXIMO

- ▷ Algoritmo:

```
CV-2-Aprox(G);      (* G = (V, E) *)
    C ← {};
    Construir um emparelhamento maximal M* em G;
    Para todo (u, v) ∈ M* faça C ← C ∪ {u, v};
Retornar C.
```

- **Teorema**: $\frac{z^H(I)}{z^*(I)} \leq 2$.

  Prova:

  Parte I: C é uma cobertura de vértices pois, se existisse uma aresta (u, v) não coberta então M^* não seria maximal.

  Parte II: $|C| \leq 2z^*(I)$.
  Se C' e M' são respectivamente uma cobertura e um emparelhamento qualquer de G então $|C'| \geq |M'|$. Logo:

$$z^*(I) \geq |M^*| = \frac{|C|}{2} = \frac{z^H(I)}{2}.$$

□

α-Aproximação

- Exemplo 2: *bin packing* unidimensional.

  Dados n arquivos de tamanhos $\{t_1, \ldots, t_n\}$ e disketes de capacidade de armazenamento C, qual o menor número de disketes necessários para fazer o *backup* de todos os arquivos ?

  Observação: supor que $t_i \leq C$ para todo $i = 1, \ldots, n$.

- Algoritmo básico:

```
Bin-Aprox(t, n, C);
    Preprocessamento(t, n);      Disketes-em-uso ← {};      k ← 0;
    Para i = 1 até n faça
        j ← Escolher-diskette(Disketes-em-uso, i);
        Se j = 0 então    (* arquivo não cabe nos disketes em uso *)
            k++;    Disketes-em-uso ← Disketes-em-uso ∪ {k};    j ← k;
        fim-se
        Armazenar(i, j);
    fim-para
Retornar k.
```

α-Aproximação

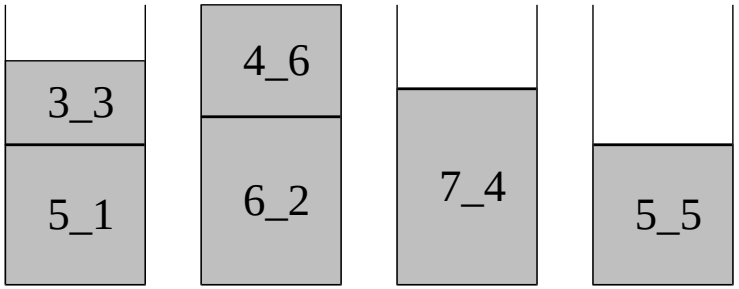
- Descrição dos procedimentos do algoritmo `Bin-Aprox`:
  - `Preprocessamento`: retorna uma nova permutação dos arquivos.
  - `Escolher-diskete`: retorna o número do diskete em uso onde será armazenado o arquivo *i* ou *zero* caso não encontre diskete com capacidade residual de armazenamento suficiente.
  - `Armazenar`: registra que o arquivo *i* será alocado ao *j*-ésimo diskete, atualizando a sua capacidade residual de armazenamento.

α-Aproximação

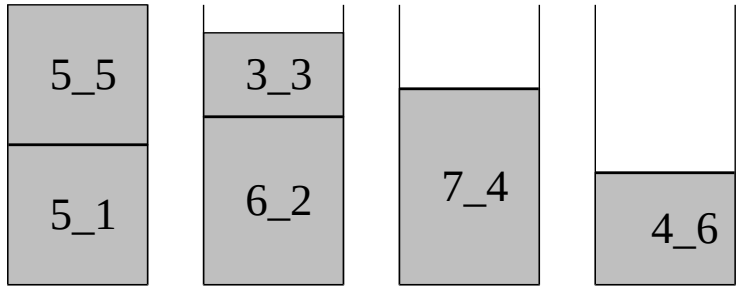
- Estratégias alternativas:
  - *First Fit* (FF): `Preprocessamento` mantém ordem dos arquivos de entrada e `Escolher-diskete` procura o diskete em uso de *menor índice* aonde cabe o arquivo corrente.
  - *Best Fit* (BF): `Preprocessamento` mantém ordem dos arquivos de entrada e `Escolher-diskete` procura o diskete em uso de *menor capacidade residual de armazenamento* aonde cabe o arquivo corrente.
  - *First Fit Decrease* (FFD): variante do algoritmo FF onde o `Preprocessamento` ordena os arquivos em ordem decrescente de tamanho.
  - *Best Fit Decrease* (BFD): variante do algoritmo BF onde o `Preprocessamento` ordena os arquivos em ordem decrescente de tamanho.

α-Aproximação

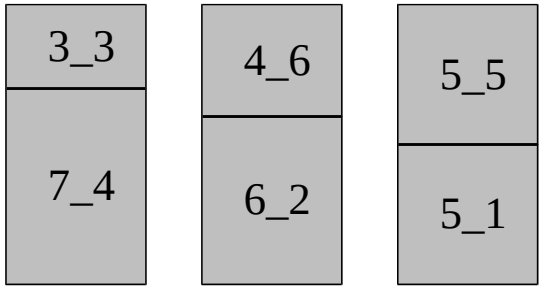
▷ Exemplo de aplicação dos algoritmos para *bin packing*:
$C = 10,\ n = 6,\ t = \{5_1, 6_2, 3_3, 7_4, 5_5, 4_6\}$ (notação: $i_j \Longrightarrow t_j = i$).



FF



BF



FFD = BFD = OTIMO

- **Teorema**: FF é um algoritmo 2-aproximado para *bin packing*.

  Prova: seja b o valor retornado por FF e b^* o valor ótimo. Suponha que os disketes estão ordenados decrescentemente pela sua capacidade residual. Note que a capacidade residual dos $b-1$ primeiros disketes da solução de FF é $\leq C/2$. Caso contrário, se dois disketes tivessem capacidade residual $\geq C/2$ os seus arquivos teriam sido armazenados em um único diskete. Como o total armazenado no diskete b é maior que a capacidade residual dos demais disketes, tem-se

$$S = \sum_{i=1}^{n} t_i \geq b\,\frac{C}{2}.$$

  Como $b^* \geq \lceil \frac{S}{C} \rceil \geq \frac{S}{C}$, a equação acima implica que $b^* \geq \frac{1}{2}\,b$. $\square$

- **Teorema**: para toda instância I do *bin packing* tem-se que

$$z^{xx}(I) \leq \frac{17}{10}\, z^*(I) + 2 \quad \text{e} \quad z^{xxD}(I) \leq \frac{11}{9}\, z^*(I) + 2,$$

onde $xx \in \{FF, BF\}$.

- **Teorema**: Não existe uma α-aproximação para TSP (genérico) com complexidade polinomial a menos que $\mathcal{P} = \mathcal{NP}$.

  Prova: Suponha que $\mathcal{P} \neq \mathcal{NP}$ e que existe um algoritmo polinomial H tal que $\frac{z^H(I)}{z^*(I)} \leq \alpha \in \mathbb{Z}_+$.
  Seja G o grafo dado como entrada do problema de decisão do ciclo hamiltoniano (HAM). Construa o grafo G' completando com as arestas que faltam. Atribua custo um às arestas originais e custo αn àquelas que foram inseridas no passo anterior.
  Se G tem um ciclo hamiltoniano, então o valor ótimo do TSP é $z^*(G) = n$. Como H é α-aproximado para o TSP

$$\frac{z^H(G)}{z^*(G)} \leq \alpha \Rightarrow z^H(G) \leq \alpha\, n.$$

- ▷ *Prova* (cont.):

  Assim, quando G tem um ciclo hamiltoniano, o ciclo encontrado por H para o TSP só terá arestas originais de G !

  Por outro lado, se G não tem ciclo hamiltoniano, $z^H(G) \geq 1 + \alpha\ n$.

  Portanto, G tem um ciclo hamiltoniano se somente se $z^H(G) \leq \alpha\ n$, ou seja, H resolve HAM em tempo polinomial.

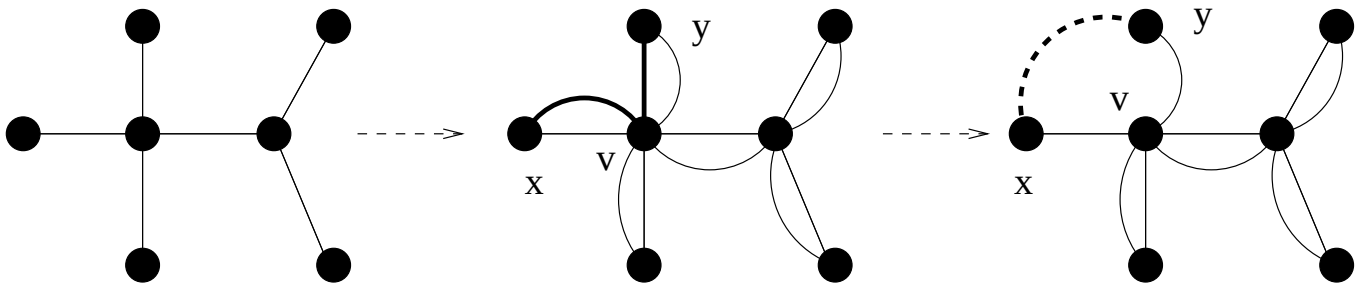
- ▷ Absurdo, já que, por hipótese, $\mathcal{P} \neq \mathcal{NP}$. □

α-Aproximação

▷ Exemplo 3: 2-aproximação para o TSP-*métrico*, ou seja, quando as distâncias obedecem à *desigualdade triangular*.

TSP-Aprox(G); (* $G = (V, E)$, $|V| = n$ e completo *)
  Construir T, uma árvore geradora minima de G;
  Construir o grafo G' duplicando-se todas as arestas de T;
  Construir um ciclo Euleriano C' em G'; (* graus de todos vértices é par *)
  Seja $\{v_1, \ldots, v_{2n-2}, v_1\}$ a ordem de visitação dos vértices em C';
  $C \leftarrow \{\ \}$;
  Para $i = 1, \ldots, 2n-2$ **faça**
    Se v_i não está marcado **então**
      Marcar v_i e fazer $C \leftarrow C \cup \{v_i\}$; (+)
    fim-para
  $C \leftarrow C \cup \{v_1\}$;
Retorne C;

α-Aproximação



▷ **Teorema**: `TSP-Aprox` é uma 2-aproximação para o TSP-*métrico*.

Prova: se z^* é o custo mínimo de um ciclo hamiltoniano em G,

$$\text{custo}(T) \leq z^* \Rightarrow 2\,\text{custo}(T) \leq 2z^*.$$

Por outro lado, devido aos custos obedecerem à desigualdade triangular, o comando $(+)$ só pode diminuir o custo de C ao longo das iterações. Logo

$$\text{custo}(C) \leq 2\,\text{custo}(T) \leq 2z^*. \qquad \square$$

Algoritmo de Christofides

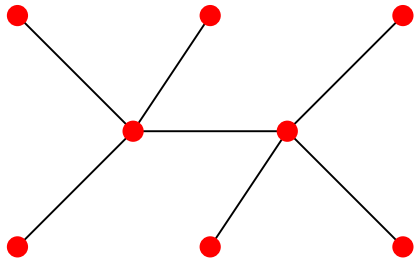
Estratégia:

Aumentar a árvore apenas o suficiente para encontrar ciclo euleriano.

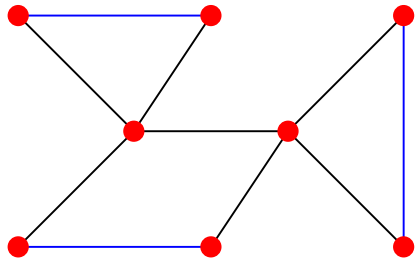
Teorema: *O problema de encontrar um emparelhamento perfeito de peso mínimo pode ser resolvido em tempo polinomial.*

TSPM-CHRISTOFIDES (G, c)

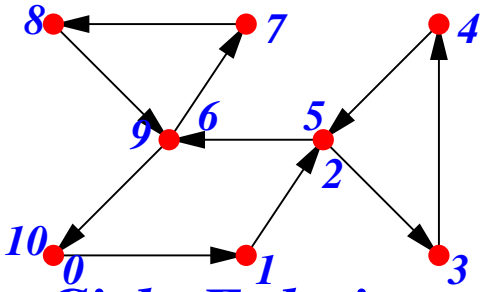
1 $T \leftarrow$ MINIMUM-SPANNING-TREE (G, c)
2 seja I o conjunto dos vértices de grau ímpar de T
3 $M \leftarrow$ EMPARELHAMENTO-PERFEITO-MÍNIMO $(G[I], c)$
4 $T' \leftarrow T \dot{+} M$
5 $P \leftarrow$ CICLO-EULERIANO (T')
6 $C \leftarrow$ ATALHO (P)
7 devolva C



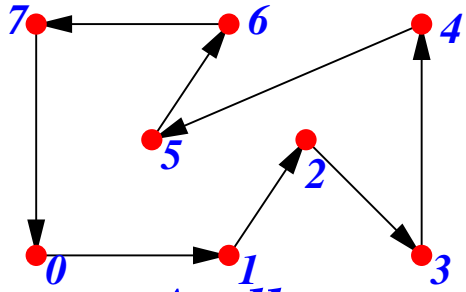
T



T+M



Ciclo Euleriano



Atalho

Teorema: TSPM-*Christofides é uma* $\frac{3}{2}$*-aproximação.*

Prova. Um circuito em I define dois emparelhamentos alternantes em I.

Assim, $c(M) \leq \frac{1}{2}\mathrm{OPT}(G)$.

Portanto $c(C) \leq c(T) + c(M) \leq \mathrm{OPT}(G) + \frac{1}{2}\mathrm{OPT}(G) = \frac{3}{2}\mathrm{OPT}(G)$. □