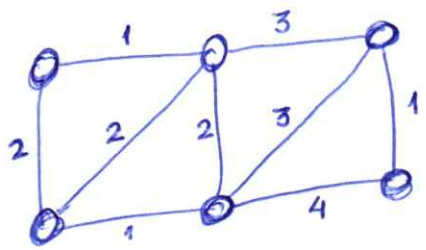
Aula 8 Algoritmos Gulosos

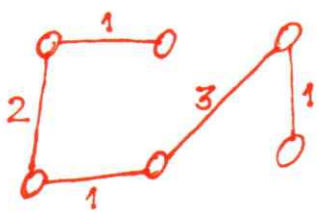
- Alguns problemas só podem ser resolvidos "olhando-se para a frente". Por exemplo, no xadrez, se você pensar somente na vantagem imediata, provavelmente vai perder!
- Já em outros problemas podemos tomar decisões tendo em vista somente o que é melhor no momento, sem se preocupar com o futuro.

- Algoritmos gulosos quase sempre são muito simples.
- Um exemplo interessante de problema com boas soluções gulosas é o de encontrar ÁRVORE GERADORA MÍNIMA. (MINIMUM SPANNING TREE)

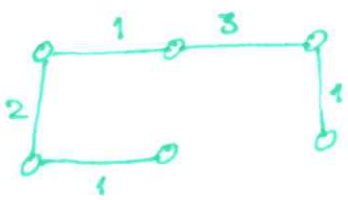
- Seja $G(V,E)$ um grafo conexo, em que cada aresta $e=(v,w)$ possui um peso $d(e)$. Denomina-se peso de uma árvore geradora $T(V,E_T)$ de G à soma dos pesos das arestas de G que formam T. Ou seja, o peso de T é $\sum_{e \in E_T} d(e)$.

- O problema é obter a árvore geradora de peso mínimo, para um dado grafo.





peso 8



peso 8

- Queremos um subconjunto $E_T \subseteq E$ tal que
  - (V, E_T) seja uma árvore
  - $\sum_{(v,w) \in E_T} d(v,w)$ seja mínimo

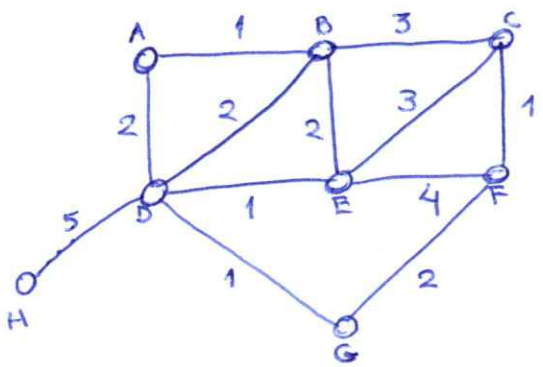
Algoritmos para encontrar árvores geradoras mínimas usando método guloso

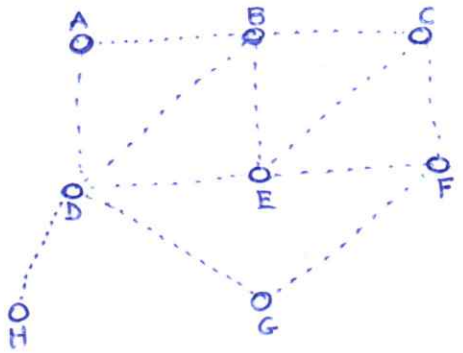
- Kruskal — Ano 1956 — Complexidade $O(m \log n)$
- Prim — Ano 1957 — Complexidade $O(n^2)$, $O(m \log n)$, $O(m + n \log n)$ ... depende da estrutura de dados usada.

Algoritmo de Kruskal

- Definir inicialmente $E_T = \emptyset$
- A cada passo, escolher uma aresta ainda não considerada, (v, w), tal que
  - inclui (v, w) não produz ciclo
  - de todas as arestas possíveis (ainda não consideradas), (v, w) é aquela que possui peso mínimo.
- Incluir (v, w) em E_T
- Repetir até considerar todas as arestas

- Vejamos um exemplo:





- Usando o "ALGOL" convencionado no livro:

Algoritmo 5.1

dados: grafo $G(V,E)$ conexo, $V=\{v_1,\dots,v_n\}$

definir conjuntos $S_j := \{v_j\}$, $1 \le j \le n$

definir $E_T = \emptyset$

$e_1, e_2, \dots, e_m := \text{ordena}(E)$

para $j := 1, \dots, m$ efetuar

  seja v, w os vértices extremos de e_j

  se v, w pertencem respectiv. a conjuntos S_p, S_q disjuntos então

    $S_p := S_p \cup S_q$

    eliminar S_q

    $E_T := E_T \cup \{e_j\}$

- Começa com uma floresta que vai sendo fundida até encontrar a solução.
- É possível (e relativamente simples) provar que o algoritmo é correto:

Lema 5.1 Seja $G(V,E)$ um grafo conexo. Seja $T(V, E_T)$ o subgrafo obtido pela aplicação do algoritmo guloso acima. Então T é uma árvore geradora de G.

Prova. - Por construção, T obtido é subgrafo gerador acíclico.

- Vamos provar que T é conexo. Suponha que não seja. Sejam T_1, T_2 duas árvores distintas da floresta T. Seja (v,w) a primeira aresta ordenada tq v está em T_1 e w está em T_2. Ao adicionar a aresta, não produz ciclo. Logo, (v,w) não poderia ser rejeitada pelo algoritmo. ∎

Lema 5.2 Seja $G(V,E)$ um grafo conexo e $T(V,E_T)$ a árvore geradora obtida pela aplicação do algoritmo. Então T possui peso mínimo.

Prova. - Sejam $e_1, e_2, \dots, e_{n-1}$ as arestas ordenadas de T, ou seja, $d(e_1) \leq \dots \leq d(e_{n-1})$.

- Suponha, por contradição, que T não seja mínima.
  - Seja $T'(V, E_{T'})$ a MST que contém as arestas $e_1, e_2, \dots, e_j$ para o maior j dentre todas as MSTs possíveis. Note que $j < n-1$, concordam?

Então nossa árvore T começou a ficar diferente de T' a partir de e_{j+1}.

- Adicione e_{j+1} à árvore T'. Vai formar um ciclo, concordam?

- O ciclo formado em T' ao adicionar e_{j+1} contém uma aresta $x \neq e_i$, $1 \leq i \leq j+1$, caso contrário não formaria um ciclo.
  Já sabemos que $e_1, e_2, \dots, e_{j+1}$ não formam ciclo, pois estavam em T.
- Sabemos que $d(e_{j+1}) \leq d(x)$, caso contrário a aresta x teria sido escolhida pelo algoritmo em vez de e_{j+1}.
  Note que $e_1, e_2, \dots, e_j, x$ não formam ciclo, pois todas essas arestas estavam em T'. Então se o algoritmo selecionou e_{j+1} em vez de x ao formar T, é porque $d(e_{j+1}) \leq d(x)$.
- Se $d(e_{j+1}) < d(x)$, contradição!
  Pois árvore $T''(V, [E_{T'} - \{x\}] \cup \{e_{j+1}\})$ possui peso menor que T'

- Se $d(e_{j+1}) = d(x)$, contradição!

Pois T'' também é mínima e contém as arestas $e_1, e_2, \dots, e_j, e_{j+1}$.

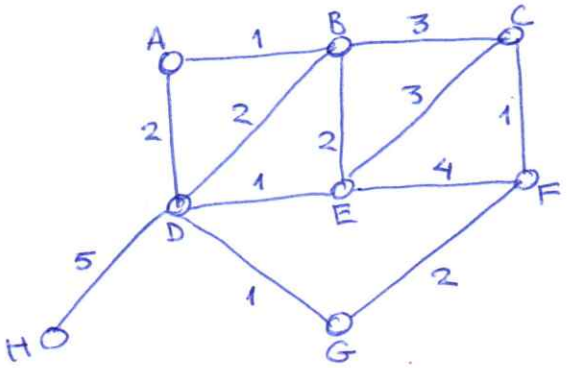
- Logo, T deve ser mínima. ∎

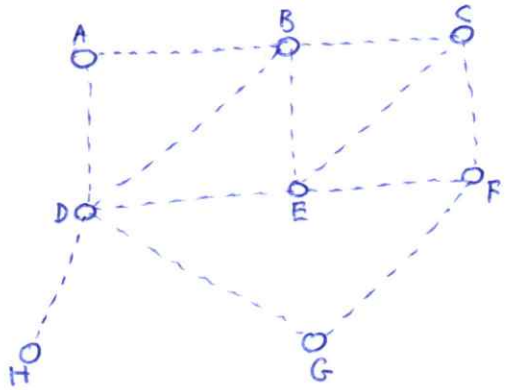
Algoritmo de Prim

- Variação do algoritmo guloso de Kruskal.
- Determina uma árvore geradora mínima $T(V, E_T)$ de um grafo $G(V,E)$
- No primeiro passo, inclua em E_T a aresta de menor peso de E.

  Depois, repita inclua em E_T a aresta de menor peso de E que possua exatamente um extremo incidente a alguma aresta já incluída em E'_T.

- Vejamos um exemplo:





Códigos de Huffman

- Usados para compressão de dados
- Utiliza tabela de frequências de ocorrências dos caracteres.

Símbolos mais frequentes → codificação com menos bits

Exemplo:

Considere um alfabeto formado somente pelas letras A, B, C, D, R, que em um certo idioma são utilizadas com a frequência abaixo

A	B	C	D	R
40%	20%	10%	10%	20%

Um código binário de comprimento fixo poderia representar as letras como, por exemplo

A	B	C	D	R
000	001	010	011	100

Um código binário de comprimento variável poderia tirar vantagem do fato de que algumas letras são mais frequentes

A	B	C	D	R
0	100	1010	1011	11

Código livres de prefixos!

- Digamos que queremos codificar a palavra

"ABRACADABRA"

Com código de compr. fixo:

000 001 100 000 010 000 011 000 001 100 000

Com código de compr. variável:

0 100 11 0 1010 0 1011 0 100 11 0

- Quando a frequência dos caracteres não for uniforme, o código de comprimento variável pode ser mais econômico.

Quão econômico?
Por que? > - Ver artigo famoso de C. Shannon!

- Entropia de Shannon: mede quantidade de informação

$$H(X) = -\sum_i p(x_i) \log_2(p(x_i))$$

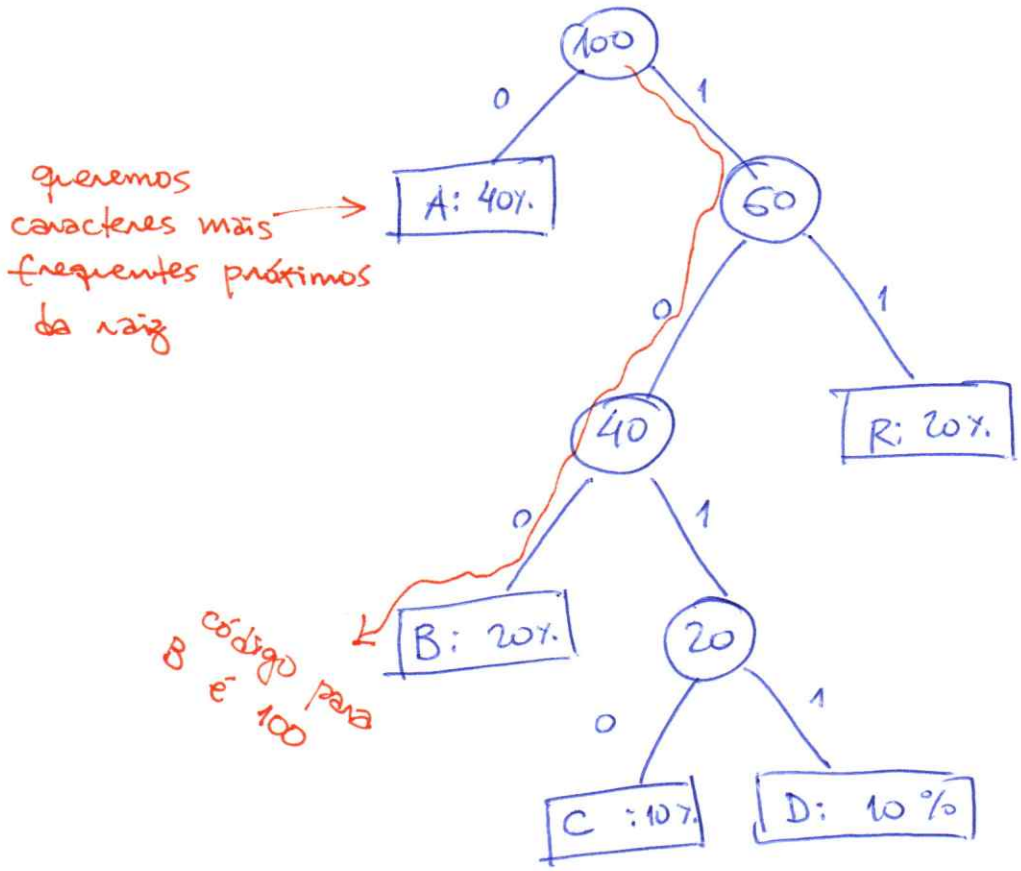
- Se um texto tem muita entropia ele pode ser pouco comprimido.

- Como gerar uma codificação ótima: ALGORITMO DE HUFFMAN

Ideia: vamos construir uma árvore

- folhas representam caracteres
- nós internos representam frequências
- arestas representam 0 ou 1.
- folhas não precisam aparecer ordenadas

No exemplo anterior:



Problema enunciado de modo mais preciso:

Encontrar uma árvore cujas folhas correspondam a símbolos e que minimize a função

$$C(T) = \sum_{i=1}^{n} f_i \cdot d_T(i)$$

f_i: frequência do i-ésimo símbolo

$d_T(i)$: profundidade do i-ésimo símbolo na árvore

Algoritmo guloso: constrói a árvore a partir das folhas de menor frequência:

Exemplo:

Primeiro passo: