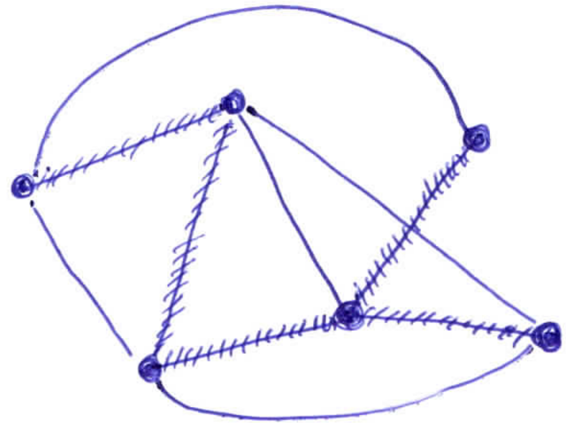
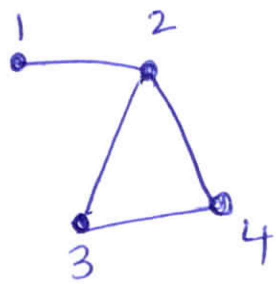
Minimum Spanning Tree



Spanning tree

Graph $G(V,E)$. Undirected.

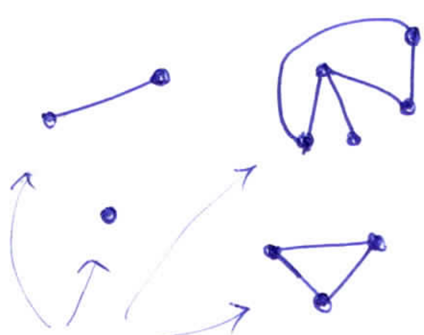


$V = \{1, 2, 3, 4\}$.

$E = \{(1,2), (2,3), (3,4), (2,4)\}$

- n vertices. Number of edges $\leq \binom{n}{2}$.
- Path. "walk" if vertices allowed to repeat.
- (simple) cycle.
- connected.

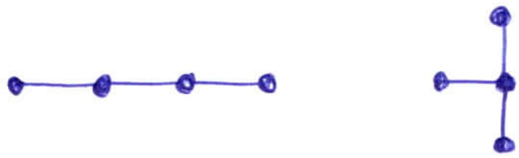
Graph that is not connected.



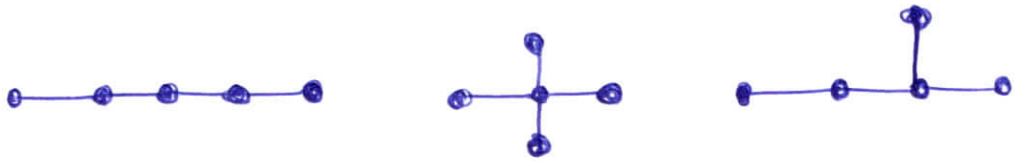
(conn) components

Def. A <u>tree</u> is a <u>connected</u> graph with <u>no cycles</u>.

$n = 4$



$n = 5$



Note A tree with n vertices has $n-1$ edges

Def A <u>spanning tree</u> of a connected graph $G(V,E)$ is a graph

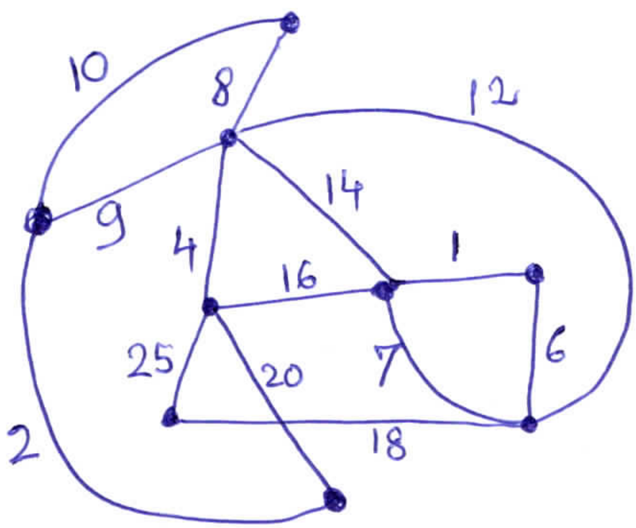
- $T(V, E')$ such that
- $E' \subseteq E$ <u>and</u> T is a tree.

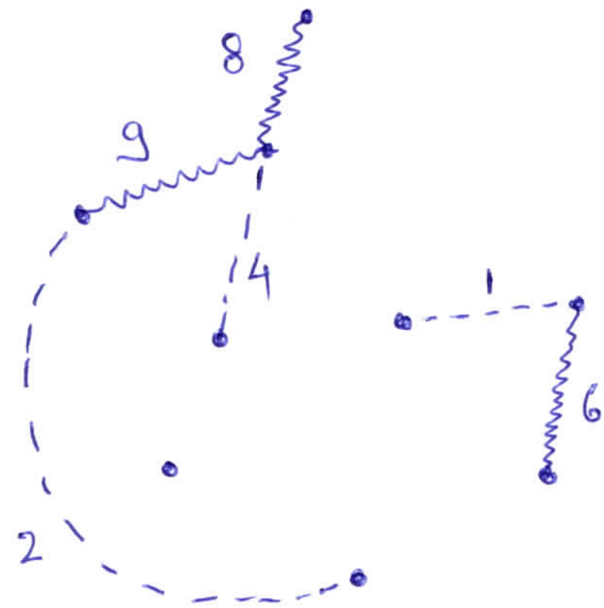
<u>Minimum Spanning Tree Problem</u>

- Given a graph $G(V,E)$ and
- for each edge $(u,v) = e$, cost $c_e \geq 0$,
- Find a spanning tree T w/ min cost.

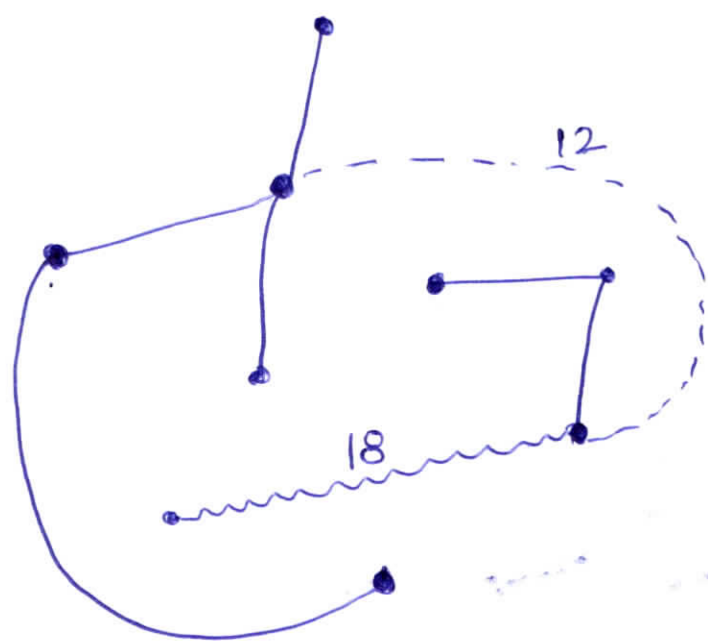
$$\text{cost}(T) = \sum_{e \in T} c_e .$$

Idea Start with empty graph. Add edges one by one, starting with min cost edge, without introducing cycles.





[Skip 7, 10].



Done!

[Ignore 20, 25].

[skip 14, 16]

Algorithm Let m = # edges.

- Sort edges according to their cost.

$$\begin{array}{ccccc} e_1 & e_2 & e_3 & \cdots & e_m \\ c_1 \le & c_2 \le & c_3 & \cdots & \le c_m \end{array}$$

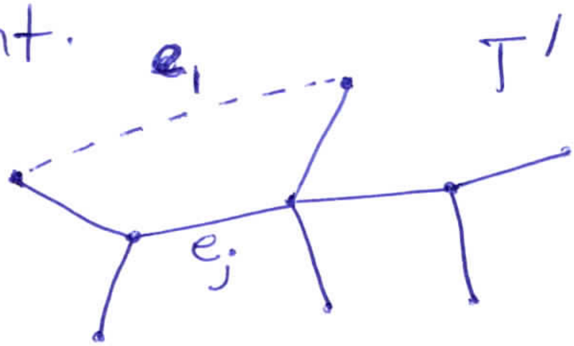
- Start with graph H with no edges.
- For $i = 1, 2, 3, \ldots, m$,
  - Add e_i to H if it does not introduce a cycle.

Theorem The algo. produces a M.S.T.

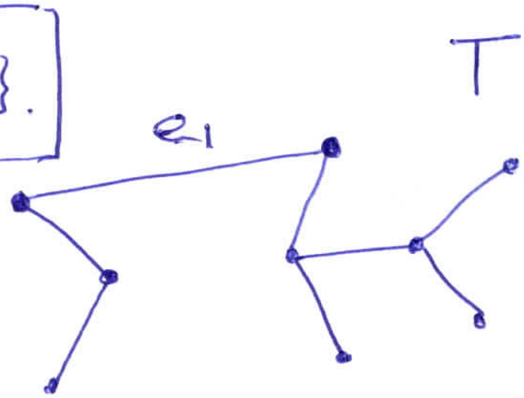
Proof idea We show that for every i, decisions made up to i^{th} step are correct in the sense that there exists a (hypothetical) M.S.T. T that among the edges $\{e_1, e_2, \ldots, e_i\}$ includes precisely those edges that are selected by the algorithm.

<u>Warm-up</u> $i=1$. Since algo. selects e_1, we need to show that $\exists$ M.S.T. T such that $e_1 \in T$.

<u>Proof</u> Exchange argument.



- Let T' be hypothetical M.S.T.
- If $e_1 \in T'$, done.
- So assume $e_1 \notin T'$.
- $\therefore$ $T' \cup \{e_1\}$ contains a cycle. Let e_j, $j > 1$ be any other edge on the cycle.
- Note $\text{cost}(e_1) \leq \text{cost}(e_j)$.
- Let $\boxed{T = T' \cup \{e_1\} \setminus \{e_j\}.}$



- $\text{cost}(T) \leq \text{cost}(T')$
  $=$
- $\therefore$ T is optimal (M.S.T.) <u>and</u> contains e_1. Done!

Formal claim Let $A_i = \{e_1, e_2, \dots, e_i\}$.

Let S_i = Set of edges selected by algo from A_i (i.e. after examining $e_1, \dots, e_i$)

Then there exists (hypothetical) M.S.T. T such that $T \cap A_i = S_i$.

This holds for $0 \le i \le m$.

Note Setting $i = m$, one concludes that the algo. outputs a M.S.T.

$$\underbrace{S_m}_{\text{Algo's output}} = T \cap A_m = \underbrace{T}_{\text{M.S.T.}}.$$

Proof $i = 0$ $\quad T \cap \phi = \phi$. Nothing to prove.

Inductive Suppose that for $i \le m-1$, there is M.S.T. T' s.t.

$$T' \cap \underbrace{\{e_1, e_2, \dots, e_i\}}_{A_i} = S_i.$$

Consider the edge e_{i+1}.

<u>case 1</u> e_{i+1} introduces an ~~edge~~ cycle in $H(V, S_i)$

- $\therefore$ Algo. skips e_{i+1}, $S_{i+1} = S_i$
- $S_i \subseteq T'$, T' is a tree, $\therefore e_{i+1} \notin T'$.

$\therefore e_{i+1}$ is neither in S_{i+1} nor in T'.

$$\therefore \quad T' \cap \underbrace{\{e_1, \ldots, e_{i+1}\}}_{A_{i+1}} = S_{i+1}.$$

(Same M.S.T. T' works). ∎

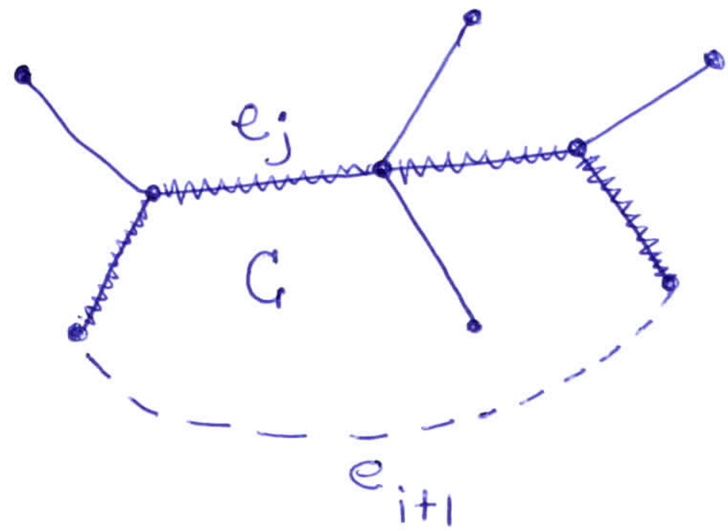
<u>Case 2</u> e_{i+1} does not introduce a cycle in $H(V, S_i)$.

$\therefore$ Algo selects e_{i+1}, $S_{i+1} = S_i \cup \{e_{i+1}\}$.

<u>Sub-case (I)</u> If e_{i+1} appears in T' as well, then same T' works,

$$T' \cap \underbrace{\{e_1, \ldots, e_{i+1}\}}_{A_{i+1}} = S_{i+1}.$$

<u>Sub-case (2)</u> e_{i+1} does not appear in T'.



$T' \supseteq S_i$

- Let C be the cycle in $T' \cup \{e_{i+1}\}$.
- Since C is not contained in $S_i \cup \{e_{i+1}\}$

otherwise algo. would not select e_{i+1},

- C must contain an edge e_j s.t. $j > i+1$.

<u>and</u> hence $cost(e_j) \geq cost(e_{i+1})$.

- $\therefore$ $T = T' \cup \{e_{i+1}\} \setminus \{e_j\}$

is M.S.T. with

$T \cap \underbrace{\{e_1, \ldots, e_{i+1}\}}_{A_{i+1}} = S_{i+1}$. Done. $\boxtimes$

Implementation (Kruskal's Algorithm)

- Keep connected components as sets.
- (Union-Find data structure).
    - Maintain sets as trees.
    - Merge (T_1, T_2) := If $|T_1| \geq |T_2|$ then make root of T_2 a child of root of T_1.
    - Exercise Prove that height (of all trees) remains $\leq O(\log n)$.
- Run-time. $O(m \cdot \log n)$.

$\blacksquare$

Huffman Codes

$a \to 10$
$b \to 111$
$c \to 100$
$d \to 001$
$e \to 01$

$abcd \longrightarrow \underline{10}\,\underline{111}\,\underline{100}\,\underline{001}$

Ambiguity.

$\underline{10001} \to ce$, $\to ad$

Def Given an alphabet $\Sigma = \{x_1, x_2, \ldots, x_n\}$ a code is a map $C : \Sigma \to \{0,1\}^*$.

Note Code must be unambiguous.

One solution
- Let $k = \lceil \log n \rceil$.
- $C : \Sigma \to \{0,1\}^k$, all encodings of same length.

However a, e occur more frequenty than b, c, d. Can we be more efficient?

Alternate C is prefix-free, i.e. for $i \neq j$ $C(x_i)$ is not a prefix of $C(x_j)$.

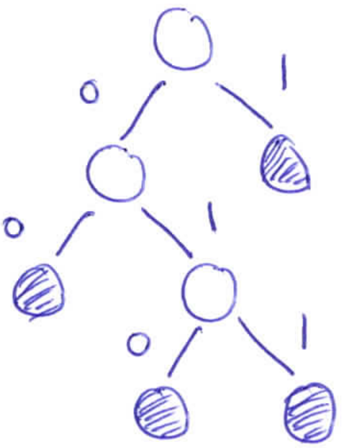
$\Rightarrow$ Unambiguous.

Problem Given $\Sigma = \{x_1, x_2, .., x_n\}$
and frequencies $f_1\ f_2 \cdots f_n$

Find a prefix-free code $c : \Sigma \to \{0,1\}^*$
so as to minimize $\sum_{i=1}^{n} |c(x_i)| \cdot f_i$.

Prefix-free codes $\equiv$ Binary trees

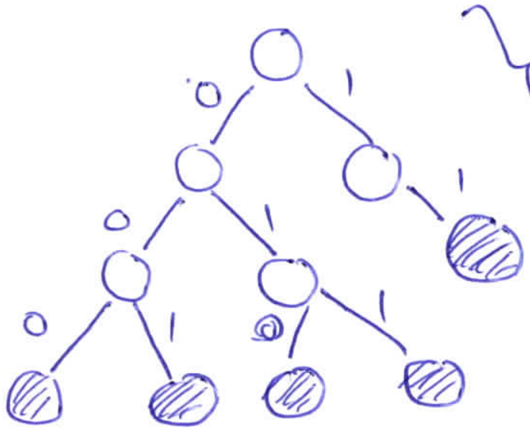
Binary tree $\Rightarrow$ Prefix-free code.



- 1
- 00
- 010
- 011

Prefix-free code $\Rightarrow$ Binary tree

11, 011, 010,
001, 000



could have shrunk to 1

- w.l.o.g. we restrict to full binary tree i.e. every node is either a leaf or has two children.

<u>Problem</u> Find a Full Binary Tree with

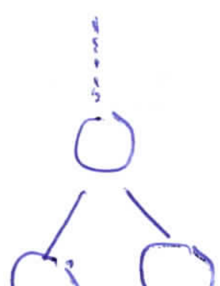
- n leaves
- A bijection from $\overset{\{x_1, \ldots, x_n\}}{\overset{\shortparallel}{\Sigma}}$ to set of leaves.
- Minimize $\sum_{i=1}^{n} f_i \cdot \text{depth}(\text{leaf of } x_i)$.

<u>Question</u> If the tree is given, what is optimal assignment of characters to leaves?

<u>clear</u> Arrange leaves in <u>decreasing</u> order of <u>depth</u>

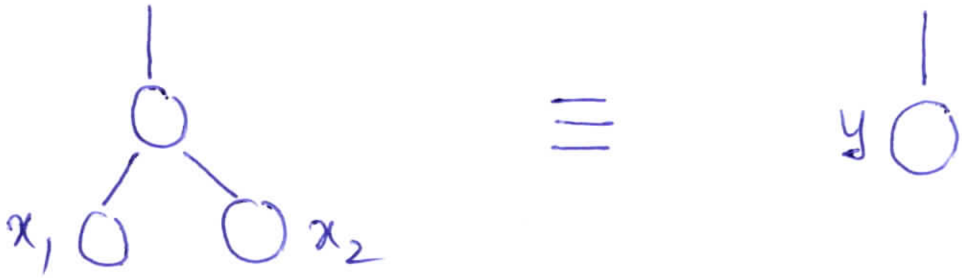
〃 characters in <u>increasing</u> 〃 <u>frequency</u>

<u>observation</u>



- There are two leaves that are siblings & at largest depth.
- w.l.o.g. they can be assigned characters with lowest frequencies

Lemma



Let $I = \{(x_1, f_1), (x_2, f_2), \ldots, (x_n, f_n)\}$ be an instance of P.F.C. problem with $f_1 \le f_2 \le \cdots \le f_n$. Let

$$J = \{(y, f_1+f_2)\} \cup \{(x_3, f_3), \ldots, (x_n, f_n)\}$$

be a new instance. Then

$$OPT(I) = OPT(J) + f_1 + f_2.$$

Proof Exercise! $\geqslant$, $\leqslant$

Algorithm Exercise!