- shortest path in a graph
- Huffman codes
- activity selection
- minimum spanning tree etc.

Activity selection problem

Input: Set of n activities
$A = \{a_1, a_2, \ldots, a_n\}$
Each activity has a start time s_i and a finish time f_i $[s_i, f_i)$

Output: max size subset of A that are mutually compatible
a_i & a_j can be scheduled together iff they do not overlap
i.e., $(f_i \leq s_j)$ OR $(f_j \leq s_i)$

Assume that the activities are in ascending order of f_i
$f_1 \leq f_2 \leq f_3 \leq \ldots\ldots \leq f_n$

Let 'l' be the no. of the last added activity

First add a_1 to S

$S \longleftarrow \{a_1\}$

$l \longleftarrow 1$

for $i = 2$ to n

{

if ($s_i \geq f_k$)
{
  $S \leftarrow S \cup \{a_i\}$
  $k \leftarrow i$
}

}
output S.

algo has greedy-choice-property.
(i.e., there exists atleast one optimal solution with a_1)

Proof: suppose there's a set B without a_1

$B = \{a_{i_1}, a_{i_2}, \ldots a_{ik}\}$

consider another set B'

$B' = (B \setminus \{a_{i_1}\}) \cup \{a_1\}$

[Remove a_{i_1} and add a_1]

No. of elements remain same

Now $f_1 \leq f_{i_1}$ since a_1 was the first element.

$f_{i_1} \leq s_{i_2}$

$\Rightarrow f_1 \leq s_{i_2}$

$\therefore$ B' is valid.

since B is the max. size, B' must also be max. size subset.

Now we have to show that algorithm has optimum substructure property. (show that greedy choice property can be applied by induction)

Consider A' that is the set of all a_i such that a_i does not overlap with a_1.

$A' = \{a_i \mid a_i$ doesn't overlap with $a_1\}$

If S' was optimum solution for A'. and should be a solution S where S is the solution $= S' \cup \{a_1\}$

If it were not i.e.,

$$\{a_1, \underbrace{\ldots\ldots}\}$$

If S' doesn't lie here, either its elements are not compatible or there are more elements (contradiction)

Huffman codes

Consider a string abbaaabbcbbbb we want to store in memory or encode through a channel in least number of bits.

We would normally encode it like this:

a : 00
b : 01
c : 10
d : 11

a : 10
b : 0
c : 110
d : 111
(make sure nothing is a prefix)
This would take 25 bits.

The frequency of occurrence for this string is

S	f
a :-	4
b :-	8
c :-	1
d -	2

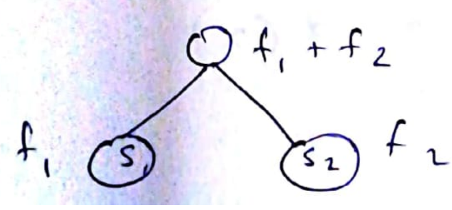
Now for any general string

s_1 - f_1
s_2 - f_2
⋮
s_n - f_n

Now store them in increasing order of frequencies $f_1 \leq f_2 \leq \ldots \leq f_n$

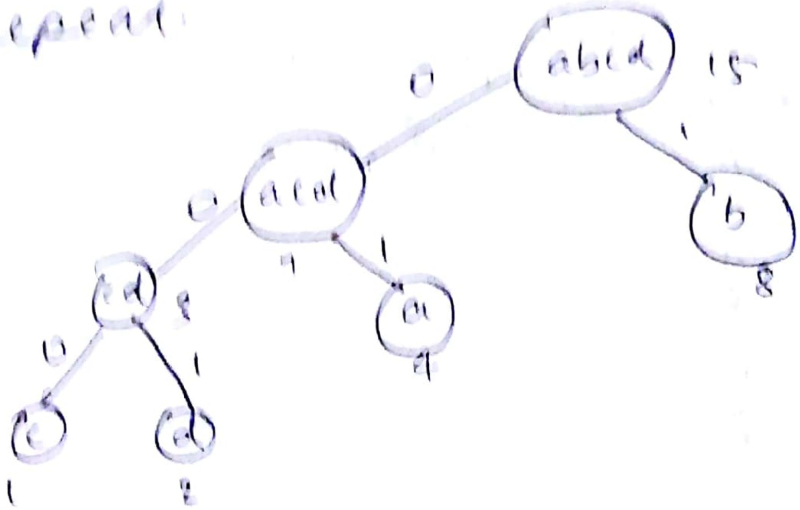
eg. c (1)
d (2)
a (4)
b (8)

Construct a Huffman tree that is a fully binary.
Start with 2 lowest frequencies make them leaves and create a parent that is sum of frequency



S_n : f_n

Again sort these in ascending and repeat.



Now starting with leftmost branch, start labelling branches with 0, 1. Prefix property will be taken care of since it's a tree

$\therefore$ a : 01

b : 1

c : 000

d : 001

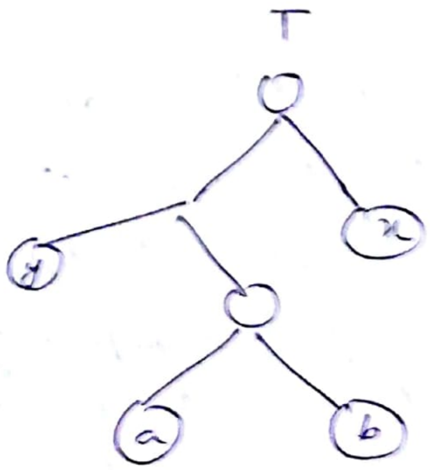
Two least frequently occuring will have maximum length

cost is the summation over all leaves of frequency multiplied by no. of bits

$$\text{Cost} = \sum_{x \in \text{leaves}} f(x) \cdot d(x)$$

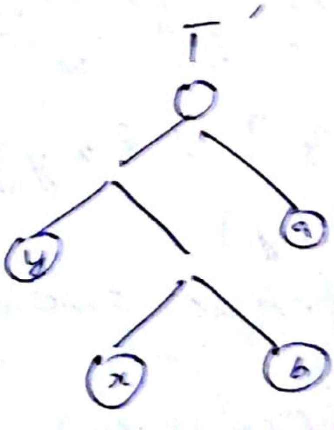
Theorem: Huffman codes have the greedy choice property i.e., if symbols x and y are the 2 least frequent ones, then ∃ an optimal tree with x & y as siblings at max. depth.

Consider another optimum tree



$f(a) \le f(b)$, $f(x) \le f(y)$

x is least frequent



Swap x and a to get T'

$\text{Cost}(T') - \text{cost}(T)$

$= f(x) \cdot d_{T'}(x) + f(a)\, d_{T'}(a)$

$- f(x) \cdot d_T(x) - f(a)\, d_T(a)$

$= f(x) \cdot d_T(a) + f(a) \cdot d_T(x)$

$- f(x) \cdot d_T(x) - f(a) \cdot d_T(a)$

$= (f(x) - f(a))\ (d_T(a) - d_T(x))$
[This is non negative since T is most optimised]

since $f(x) \leq f(a)$ [since x is least frequent], $f(x) - f(a)$ is negative or zero

However $d_T(a) - d_T(x)$ must be positive.

But since $cost(T') - cost(T)$ is non negative, either T' is optimised or a quantity is zero

construct T'' by swapping b and y and a and x. It becomes the optimum tree we are looking for

To show optimum substructure property. Let S be an symbols for optimum tree T

$S' = [S \setminus \{x, y\}] \cup \{xy\}$

$f(xy) = f(x) + f(y)$

and S' is optimal solution for T'

$$cost(T) = \sum_{x \in leaf} f(x) \cdot d(x)$$

$$cost(T') = \sum f(x) \cdot d(x)$$

$$cost(T) - cost(T') = d(x) \cdot f(x) + d(y) \cdot f(y) - d(xy) \cdot f(xy)$$

$d(xy) = d(x) - 1$

$\therefore$ cost (T) - cost (T')

$= d(x) [f(x) + f(y)] - (d(x) - 1)$
$(f(x), f(y))$

$= f(x) + f(y)$

Therefore, it is independent of depth or where x, y is.
If T' was not optimum, you can get an optimum T'' and make that optimum by adding x and y as leaves somewhere and cost will be less than T.

<u>Set-cover</u>

Given an instance of S and a set family $F = \{S_1, S_2, S_3, \ldots S_n\}$

$S_i \subseteq S$

Output: Indices $i_1, i_2, \ldots i_k$ such that $\bigcup S_{i_j} = S$. We want to minimise cost k.

Take $I = \phi$ (null set)
$U = S$ (universal set)
repeat
- choose a set S_j from F that covers the max. no. of elements in U
- $I \leftarrow I \cup \{j\}$
- $U \leftarrow U \setminus S_j$

until $U = \phi$

Scanned by CamScanner

This is the greedy approach but it will not work

$S = \{1, 2, 3, 4, 5, 6, 7\}$

$F = \{\underbrace{\{1, 2, 3, 4\}}_{S_1}, \underbrace{\{5, 1, 2\}}_{S_2}, \underbrace{\{6, 3\}}_{S_3}, \underbrace{\{7, 4\}}_{S_4}\}$

acc. to algo.,

we choose S_1 first

Then it goes on to S_2, S_3, S_4.

S_1 need ~~to~~ not belong to optimum solution, hence does not fulfil greedy choice property.

Output of greedy choice is ~~atleo~~ max. $(\log n)$ times optimum.

$$|I| \leq O(\log n |I^*|)$$

<u>Matroid theory</u>

M is a matroid

$M = \langle S, F \rangle$

Matroid is an ordered pair of set and family provided that

(i) $S \neq \phi$

(ii) Hereditary property

if $A \in F$, $\forall B \subseteq A$, $B \in F$

$\therefore \phi$ always belongs to F

(iii) Exchange property

if $A \in F$, $B \in F$, $|A| < |B|$

then $\exists x \in B \setminus A$ s.t. $A \cup \{x\} \in F$

Every element of F is called independent set.

Find a max weighted independent set

let $S = \{S_1, S_2, \ldots, S_n\}$

S_i has a weight > 0.

Minimum spanning tree (MST)

Input: undirected graph $G = (V, E)$
Edges have +ve weights

Output: min cost spanning tree of G.

Solve using matroid

Define $M_G = \langle E, F_G \rangle$

set of edges

$F_G = \{A \subseteq E \mid A \text{ is acyclic}\}$

$E \neq \emptyset$

Hereditary property is true because a subset of A cannot be acyclic.

To show exchange property:

Let $A \in F_G$
$B \in F_G$

$|A| < |B|$

Theorem: Any forest on n nodes with t trees has exactly $(n - t)$ edges

$|A| < |B|$

$(n - |A|)$ trees

$(n - |B|)$ trees

$(n - |A|) > (n - |B|)$

There must exist at least one tree T_B in $(n - |B|)$ that cuts multiple trees in $|B|$ since there are more trees in $|A|$

Let $(u,v) \in B$, $(u,v) \notin A$

$A \cup \{(u,v)\} \in F_G$

adding (u,v) to A does not make it cyclic.

$M_G = \langle E, F_G \rangle$ is a matroid

This solves max. problem.

To get solution of min, for each edge $w_i' = W - w_i$ (subtract individual weights from a large no.)

Optimum answer $A \in F$

A is maximal

$\nexists x$ s.t. $A \cup \{x\} \in F$.

All maximal independent sets are of the same size.

$M = \langle S, F \rangle$

w_i = weight of $S_i \in S$, $w_i > 0$

Sort elements of S in non increasing order of weights.

$A = \phi$

Loop Choose the next S_i (in that order) if $A \cup \{s_i\} \in F$

$A \leftarrow A \cup \{s_i\}$

Output A

A will be maximal independent set in matroid.

First show that algorithm has greedy choice property.

$\exists A$ st. $x \in A$

Let B be optimum solution, $x \notin B$.

$\phi \in F$

$B \in F$ since B is optimum soln.

Using exchange property.

~~$\exists x$~~ $\{x\} \in F$ (due to heriditary property)

If $|B|$ is 1, then $\{x\}$ is also optimal.

If $|B| > 1$, by exchange property, add elements to $\{x\}$ from B.

$\{x, y_1, y_2, y_3, \ldots y_{|B|-1}\} \in F$

until they become nearly same size. We know $w(x) \geq w(y)$ since we chose max. in the beginning.

Therefore $\{x, y_1, y_2, \ldots\}$ must be optimal.

To show optimum substructure property:

Given $x \in A$

To find rest of optimal solns.

$S' = \{y \mid y \cup \{x\} \in F\}$

$F' = \{A \in F \mid A \cap \{x\} \neq \phi\}$

consider optimum soln. A

$A' \cup \{x\} = A$

<u>set-cover (ctd.)</u>

Let I^* be the min sized set cover and I be the greedy output.

In iteration i, if n_i elements are not yet covered, it will choose a set S_i

$\frac{n_i}{|I^*|}$ elements are covered by atleast some set S_j by pigeonhole principle.

S_i covers at least $\frac{n_i}{|I^*|}$

Before starting $n_0 = n$

$$n_1 \leq n_0 - \frac{n_0}{|I^*|} = n\left(1 - \frac{1}{|I^*|}\right)$$

$$n_2 \leq n_1 - \frac{n_1}{|I^*|} \leq n\left(1 - \frac{1}{|I^*|}\right)^2$$

$$n_3 \leq n_2 - \frac{n_2}{|I^*|} \leq n\left(1 - \frac{1}{|I^*|}\right)^2$$

$\vdots$

$$n_t \le n_t \left(1 - \frac{1}{|I^*|}\right)^t$$

if $x < 1$,

$(1-x) \le e^{-x}$

$\therefore n_t \le n\left(1 - \frac{1}{|I^*|}\right)^t \le n e^{-t/|I^*|}$

greedy will stop when

$\frac{t}{|I^*|} \le \log n$

$\frac{|I|}{|I^*|} \le \log n$ (since $t = |I|$)

DYNAMIC PROGRAMMING

given a DAG with edge weights, find shortest path from u to v in the topological sorting.

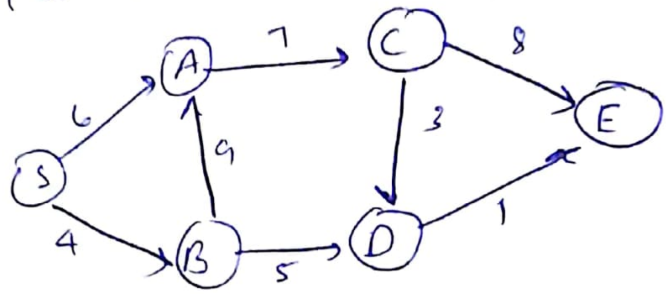
$$L(s,E) = \min_{(u,E) \in \text{Edgeset}} \{ L(s,u) + w(u,E) \}$$

i.e., for all neighbours of ending node E, the ~~sor~~ shortest path till the neighbours + weight connecting to E and take minimum of them.

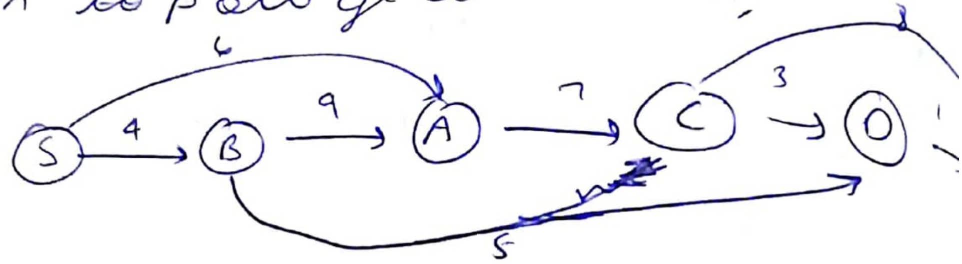
Loop In the topological sorted order,

$$L(u) = \min_{(v,u) \in E} \{ L(v) + w(v,u) \}$$

$L(S) = 0$



In topological order,



$L(B) = 4$

$L(A) = \min \begin{cases} L(S) + 6 \\ L(B) + 9 \end{cases} = 6$

$L(C) = \min \begin{cases} L(A) + 7 \\ \cancel{L(B} \end{cases} = 13$

$L(D) = \min \begin{cases} L(C) + 3 \\ L(B) + 5 \end{cases} = 9$

$L(E) = \min \begin{cases} L(C) + 8 \\ L(D) + 1 \end{cases} = 10$

<u>Longest increasing subsequence</u>

Input: sequence of nos. $a_1, a_2, a_3, \ldots, a_n$

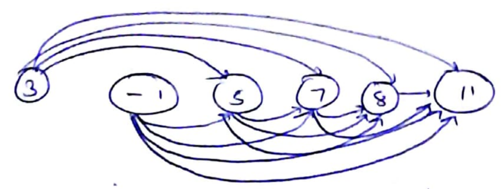
Output: $a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_k}$

$i_1 < i_2 < \ldots < i_k$

if $j > t$, $a_{i_j} > a_{i_t}$

Create a DAG

eg. 3 -1 5 7 8 11



LIS will be the longest ~~subsequen~~ path in the DAG

If suppose we want to find LIS ending at a defined a_i

$$L(i) = \max_{(j,i) \in E} \{1 + L(j)\}$$

If we don't specify an a_i, LIS will be max. of all $L(i)$s

<u>Edit distance</u>

~~types~~

given string operation edit, delete, overload

eg. EARTH

$\downarrow 2$

HEART

Edit distance is min. ~~edit~~ no. of operations required to convert.

Align the 2 strings

_EARTH
HEART_

$X = x_1 x_2 x_3 \ldots x_n$

$Y = y_1 y_2 \ldots \ldots y_m$

$E(i, j)$ = edit distance between
$x_1, x_2 \ldots x_i$
$y_1\ y_2 \ldots y_j$

$E(i, 0) = i$

$E(0, j) = j$

alignment could end as

(i) $\frac{\ \ }{y_j}$ OR (ii) $\frac{x_i}{\ \ }$ OR (iii) $\frac{x_i}{y_j}$

~~If (i) E(i, j) =~~

$$E(i, j) = \begin{cases} 1 + E(i, j-1) & \text{, in case (i)} \\ 1 + E(i-1, j) & \text{, in case (ii)} \\ E(i-1, j-1) + \text{diff}(x_i, y_j) & \text{in case (iii)} \end{cases}$$

$\downarrow$

if x_i is same as y_j, add 0, else add 1

$\text{diff}(x_i, y_j) = \begin{cases} 0, & \text{if } x_i \ldots \\ 1, & \text{otherw}\ldots \end{cases}$

	0	1	2	...		n
0	0	1	2			n
1	1					
2	2					
.						
m	m					

eg. X = EARTH

Y = HEART

		E	A	R	T	H
	0	1	2	3	4	5
0	0	1	2	3	4	5
H 1	1	1	2	3	4	4
E 2	2	1	2	3	4	5
A 3	3	2	1	2	3	4
R 4	4	3	2	1	2	3
T 5	5	4	3	2	1	$\boxed{2}$