DAA UNIT 4

LOWER BOUNDS

Estimate for minimum amt of work to solve a problem.

can be exact or efficiency class.

tight lower bound → efficiency class lower bound ⇒ algo with that class exists.

Methods:

- trivial : counting input, output
- adversary : try to make algo work as hard as possible
- problem reduction : if P reduces to Q,
  lower bound (Q) = lower bound (P)

Decision trees : comparison based sorting

For any n items, $n!$ combinations

We need at least $n!$ outcomes in decision tree.

If height h → at most h comparisons → 2^h leaves

$\Rightarrow 2^h \geq n!$

Avg comp. = Avg depth of leaves

General: $h \geq \lceil \log_2 l \rceil$

$h \geq \log_2 n!$

$\boxed{\Rightarrow h \geq n \log_2 n}$ → lower bound. } comparison

$\boxed{C_{worst}(n) = h \geq \lceil \log_2 (n+1) \rceil}$ [sorted array]

P, NP, NP-complete

$\boxed{P \subseteq NP}$

P : Polynomial : $O(p(n))$

NP : Nondeterministic Polynomial
- solved in non-polynomial
- verified in polynomial

random string → verifies

Decision ver: TSP, knapsack, graph colouring.

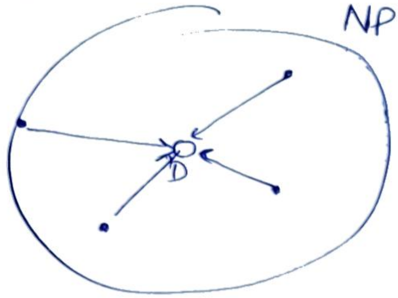
Boolean in CNF satisfiability / Hamiltonian existence

NP - Complete :

$D \in$ NP- Complete when

- $D \in NP$
- All NP $\xrightarrow[\text{time}]{\text{polynomial}}$ D

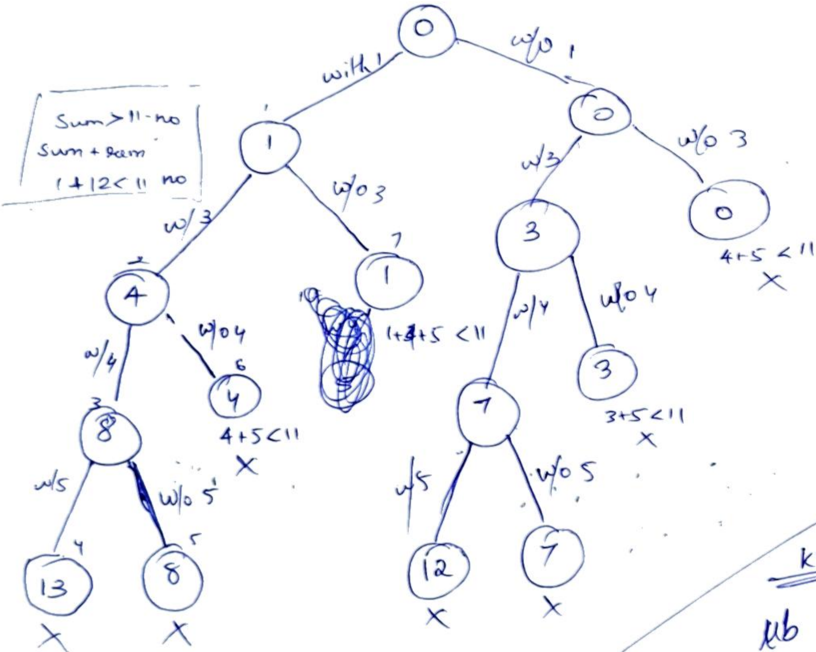
eg: CNF-sat



| if p(n) algo for NP complete algo found $\Rightarrow$ NP=P |

Backtracking

Subset sum : $A = \{1, 3, 4, 5\}$, $d = 11$



no subset

if $x[1 \ldots i] = S_i$ (soln)
write X

else
for each $x \in X$ consistent
w/ S_{i+1} &
constraints
$X[i+1] \leftarrow x$
Backtrack$(X[1 \ldots i+1])$

KNAPSACK

$ub = v + (W-w)\left(\frac{v_{i+1}}{w_{i+1}}\right)$ → largest among remaining

Branch & Bound

Assignment: initial $lb = \sum$ smallest in each row

TSP : $lb = \left\lceil \left[(\text{smallest two sum for a}) + \cdots \right] / 2 \right\rceil$

change (-,-) for a, d if including $a \xrightarrow{5} d$ as:

$(\underset{a}{-, 5})$, $(\underset{d}{5, -})$

Dynamic Programming

Set up recurrence relating soln of larger instance to soln of smaller instances.

Difference from divide & conquer : stores soln to subproblems

→ solves subproblems independently

Binomial coefficient:

Coefficients to the eqn

$$(a+b)^n = C(n,0)\,a^n b^0 + C(n,1)\,a^{n-1} b^1 + \dots + C(n,k)\,a^{n-k} b^k + \dots\, C(n,n)\,a^0 b^n$$

Recurrence:

$$C(n,k) = C(n-1,k) + C(n-1,k-1) \quad \text{for } n > k > 0$$

$$C(n,0) = 1 \quad \text{for } n \geq 0$$
$$C(n,n) = 1$$

$\Theta(nk)$

n \ k	0	1	. . .	k-1	k
0	1				
1	1	1	. . .		
:					
n-1				C(n-1,k-1)	C(n-1,k)
n					C(n,k)

Knapsack : Bottom Up

i → no of items; j → capacity

$$F(i,j) = \begin{cases} \max\left[F(i-1,j),\ v_i + F(i-1,\, j-w_i)\right] & \text{if } j - w_i \geq 0 \\ F(i-1,j) & \text{if } j - w_i < 0 \end{cases}$$

P.T.O.

item i	weight w_i	value v_i
1	2	12
2	1	10
3	3	20
4	2	15

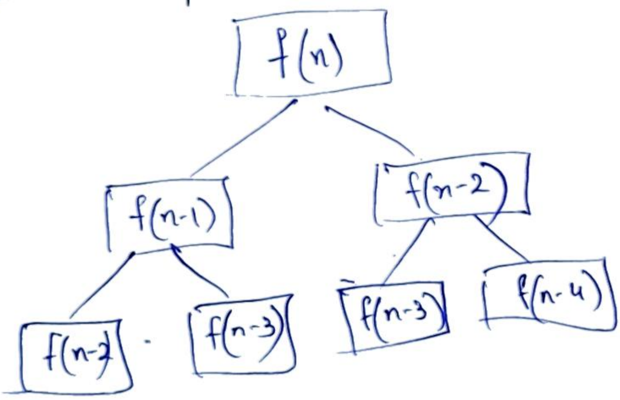
Capacity = 5.

i \ j	0	1	2	3	4	5
1		0	12	12	12	12
2		10	12	22	22	22
3		10	12	22	30	32
4		10	15	25	30	37

Advantage: each value computed only once

Disadvantage: unnecessary subproblems also calculated.

Knapsack : Top Down



$$F(i,j) = \begin{cases} \max\left[F(i-1,j),\ v_i + F(i-1, j-w_i)\right] \\ F(i-1,j) \end{cases}$$

i \ j	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	12	12	12	12
2	0		12	22		22
3				22		32
4						→ 37

$\Theta(nW)$

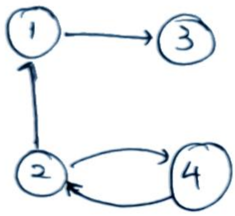
$i = 4$

$j - w_i = 5 - 2 = 3$

$i \rightarrow i-1$

WARSHALL'S

Transitive closure : node reachability



Construct matrix :

R_0:

	1	2	3	4
1	0	0	1	0
2	1	0	0	1
3	0	0	0	0
4	0	1	0	0

$\boxed{i=1}$

R_1:

	1	2	3	4
1	0	0	1	0
2	1	0	1	1
3	0	0	0	0
4	0	1	0	0

$\boxed{i=2}$

R_2:

	1	2	3	4
1	0	0	1	0
2	1	0	1	1
3	0	0	0	0
4	1	1	1	1

$\boxed{i=3}$

R_3:

	1	2	3	4
1	0	0	1	0
2	1	0	1	1
3	0	0	0	0
4	1	1	1	1

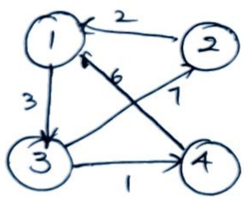
$\boxed{i=4}$

R_4:

	1	2	3	4
1	0	0	1	0
2	1	1	1	1
3	0	0	0	0
4	1	1	1	1

FLOYD'S

All pairs shortest paths : same as warshall but with weights.



Construct matrix :

R_0 :

	1	2	3	4
1	0	∞	3	∞
2	2	[illegible]	∞	∞
3	0	7	0	1
4	6	∞	∞	0

$\boxed{i=1}$

R_1 :

	1	2	3	4
1	0	∞	3	∞
2	2	0	2+3=5	∞
3	∞	7	0	1
4	6	∞	6+3=9	0

$\boxed{i=2}$

	1	2	3	4
1	0	∞	3	∞
2	2	0	5	∞
3	9	7	0	1
4	6	∞	9	0

$\boxed{i=3}$

	1	2	3	4
1	0	10	3	4
2	2	0	5	6
3	9	7	0	1
4	6	16	9	0

$\boxed{i=4}$

	1	2	3	4
1	0	10	3	4
2	2	0	5	6
3	7	7	0	1
4	6	16	9	0