Showing a problem is NPC, in 3 ways

- Decision problem vs optimization problem — NP Completeness applies directly to decision problems but not to optimization problems.
  - Path (Shortest path problem)
- Reductions — $L_1 \leq_p L_2$
  - If sol of L_2 exist, L_1 can be solved as well
- A first NP Complete problem — Circuit satisfiability problem

Instance set

Language | Certificates

The decision problem PATH has the language

$PATH = \{\langle G, u, v, k\rangle : G = (V, E)$ undirected
$u, v \in V$
$k \geq 0$ is an integer,
$\exists$ path from u, v in G
consisting of atmost k edges.$\}$

standard encoding

Abstract problem

Q.

(I, S)

binary relation on a set I of prob instances and set S of solution instances.

Counte problem

Instance set is the set of binary strings.

Encoding — affects the running time of algorithm

$\Theta(n) = \Theta(k)$ when integer k is encoded as unary

$\Theta(2^n) = \Theta(k)$ when k encoded in binary

$n = \lfloor \lg k \rfloor + 1$

Although we skip the encoding complexity when talking about Poly time computability.

Encoding in base 2 is equivalent to base 3 encoding

But UNARY encoding might change complexity

$\langle G \rangle$ denotes standard encoding which can be poly computed base 2 encoding

NP Completeness

$L \subseteq \{0,1\}^*$ is NPC if

1) $L \in NP$
2) $L_2 \leq_P L$ for every $L_2 \in NP$

NP Hard

2 \ 1 (not necessarily)

$L_2 \leq_P L$ for every $L_2 \in NP$

THM 34.4

If any NPC problem is in P, then NP = P.

Proof: Suppose $L \in P$ & $L \in NPC$.

For any $L_2 \in NP$, we have $L_2 \leq_P L$

By Lem 34.3, we get

$L_2 \in P$, since first

Next East

Lemma 34.3

$L_1 \leq_P L_2$

$L_2 \in P \Rightarrow L_1 \in P$

CIRCUIT SAT (first NP complete problem)

Proving method for NPC

1. Prove $L \in NP$
2) Select L' a known NPC problem
3) Find f $x \rightarrow f(x)$
   $L' \rightarrow L$
4) f satisfies $x \in L'$ iff $f(x) \in L$ $\forall x \in \{0,1\}^*$
5) Prove f works in polytime

NPC Tree

CIRCUIT SAT
↓
SAT
↓
3 CNF SAT
↓ ↓
CLIQUE — SUBSET SUM
↓
VERTEX COVER
↓
HAM CYCLE
↓
TSP

CIRCUIT-SAT $\leq_p$ SAT

For ex. $\phi = \big((x_1 \to x_2) \vee \neg\big((\neg x_1 \leftrightarrow x_3) \vee x_4\big)\big) \wedge (\neg x_2)$

has the satisfying assg.

$x_1 = 0,\ x_2 = 0,\ x_3 = 1,\ x_4 = 1$

Thus $\phi \in$ SAT

Requires (Naive algo) 2^n possible assgn in a ϕ formula with n variables.

Encoding $\langle\phi\rangle$ is polynomial in n.

SAT is NPC

CIRCUIT SAT $\leq_p$ SAT | SAT $\in$ NP

i) Step 1: Show SAT $\in$ NP. [Verifier poly time exist]

ii) Step 2: Prove NP Hard CIRCUIT SAT $\leq_p$ SAT

i) Given a certificate (assg of possible variables s.t. $\phi = 1$) we can verify in poly time that $\phi = 1$ or not.

So SAT $\in$ NP

ii) Method 1: For every each circuit symbol in the form of i/p input as boolean expression.

Unfortunately this reduction is not polynomial

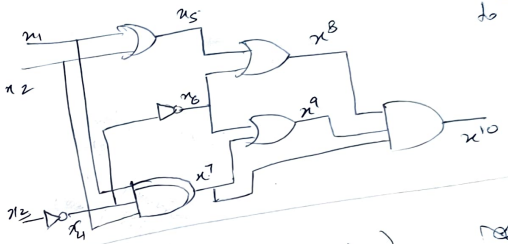
Method 2: Clever method.

- Op of o/p AND gate is

$x_{10} \leftrightarrow (x_7 \wedge x_8 \wedge x_9)$

CIRCUIT used in CIR-SAT

Reducing CIR-SAT satisfiability to formula satisfiability.



$\phi = x_{10} \wedge (x_7 \wedge x_8 \wedge x_9)$

$\wedge (x_7 \leftrightarrow (x_4 \wedge x_1 \wedge x_2))$

$\wedge (x_8 \leftrightarrow (x_5 \vee x_6))$

$\wedge (x_9 \leftrightarrow (x_6 \vee x_7))$

$\wedge (x_5 \leftrightarrow (x_1 \vee x_2))$

$\wedge (x_6 \leftrightarrow \neg x_4)$

$\wedge (x_4 \leftrightarrow \neg x_3)$

Since $x_{10} = 1$ for SAT

$x_{10} \leftrightarrow (1)$

replaced by

$x_{10} \wedge (\ )$

Thus given a circuit C, it is straightforward to prove such a formula ϕ exists in poly time.

So reduction complete.

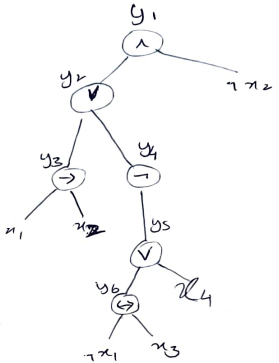
3-CNF SAT

— ϕ given in 3-CNF is satisfiable/not?

THM 34.10
Satisfiability of boolean formulas in 3-CNF is NPC.

i) To show NP, we have to show that exist a poly time verifier.

ii) NPHT: $SAT \leq_p$ 3-CNF SAT



Step 1.

Tree corresponding to formula.

$$\phi = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$$

so $\phi' = y_1 \wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \wedge (y_2 \leftrightarrow y_3 \vee y_4) \wedge (y_3 \leftrightarrow (x_1 \rightarrow x_2)) \wedge (y_4 \leftrightarrow \neg y_5) \wedge (y_5 \leftrightarrow y_6 \vee x_4) \wedge (y_6 \leftrightarrow (\neg x_1 \leftrightarrow x_3))$

3 steps

— S1: same method used in CIRCSAT $\leq_p$ SAT
y_i rep O/P
← each clause has atmost 3 literals.

— S2: convert each clause ϕ_i' into CNF
← Construct a truth table for ϕ_i' evaluating all possible assg.

introduce atmost 8 clauses $2^3 \leq 8$

— S3: Transform the formula so that each contain 3 distinct literals.

— $C_i = (l_1 \vee l_2)$ convert to $(l_1 \vee l_2 \vee p) \wedge (l_1 \vee l_2 \vee \neg p)$
p is to fulfil syntactic req.

— $C_i = (l)$ then $(l \vee p \vee q) \wedge (l \vee p \vee \neg q) \wedge (l \vee \neg p \vee q) \wedge (l \vee \neg p \vee \neg q)$

Introduce atmost x_4 clauses.

Step 2: We take $y_1 \leftrightarrow (y_2 \wedge \neg x_2)$ and convert to CNF

Convert DNF (6 results) via demorgan $(y_1 \wedge y_2 \wedge x_3) \vee (y_1 \wedge \neg y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge \neg x_2) \vee (\neg y_1 \wedge y_2 \wedge x_2)$

$A \leftrightarrow B$
$(A \wedge B) \vee (\neg A \wedge \neg B)$

$(y_1 \wedge (y_2 \wedge \neg x_2)) \vee (\neg y_1 \wedge \neg(y_2 \wedge \neg x_2))$

~~Design~~ $\phi'' =$

y_1	y_2	x_2	$(y_1 \leftrightarrow y_2 \wedge \neg x_2)$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	1

CLIQUE Problem

- Optimization problem
  of finding a CL of max size
- As a DECISION problem,
  we ask if $\exists$ Clique$_k$
- Naive algo
  - Form all subsets of size k
  - Check if a CL
  - $\rightarrow \binom{|V|}{k}$, poly if k is a constant
  - In general $k \approx \binom{|V|}{2}$ so then it runs in exponential time

CLIQUE is NPC

- Can check if $\exists$ a subset k so that it forms a clique
  NP proved
- For NP Hard

$3CNFSAT \leq_p CLIQUE_k$

Reduction

$3CNFSAT \leq_p CLIQUE_k$

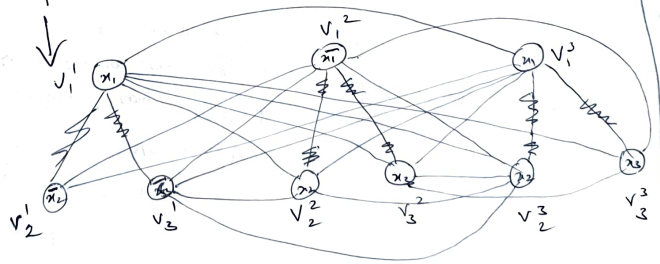
Take an instance of 3CNFSAT.

$\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ (k clauses)

$C_R = (l_1^r \vee l_2^r \vee l_3^r)$

We construct a graph G_ϕ such that ϕ is satisfiable iff G_ϕ has a clique of size k.

For $\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$



Construction of G_ϕ.

- For each C_r, we place 3 vertices.
  v_1^r v_2^r v_3^r in V
- Put an edge b/w two vertices v_i^r to v_j^s if both hold.
  i) v_i^r to v_j^s are in diff triples. $r \neq s$
  ii) ~~$\not\equiv$~~ Corresponding literal $l_i^r \neq \overline{l_j^s}$

Find function for all 0's.

$\neg\phi' = (y_1 \wedge y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge \neg x_2) \vee (\neg y_1 \wedge y_2 \wedge \neg x_2)$

$\phi' = (\quad)' = (\quad)' \wedge (\quad)' \wedge (\quad)' \wedge (\quad)'$

$= \phi''$

Pick a literal l_i^r assigned 1, corresponding to v_i^r.
Pick such literal yields a set V' of k vertices.
Claim that V' is a clique

$x_1 = 1$ $\quad x_1\ x_2\ x_3$
$\quad\quad\ 1\ 1\ 1$

$\bar{x}_1 = 1$ $\quad \bar{x}_1\ \bar{x}_2$ x_3

$\bar{x}_2\ x_3^2\ x_3^3$

$x_1 = 0, 1$ doesn't matter

VERTEX COVER

$G = V, E$

$VC = V' \subseteq V \mid \forall (u,v) \in E$
then
$u \in V'$
or
$v \in V'$
or both

~~Strategy~~

OPTIMIZATION: Find min VC
pair

DECISION: Does G have a VC of size k.

THM 34.12

VC is NPC

Proof: $G = (V, E)$, k.

Certificate: VC $V' \subseteq V$ itself.

Verification alg confirms $|V'| = k$
& checks for each edge $(u,v) \in E$ that $u \in V'$ or $v \in V'$] polytime $\Rightarrow$ NP

P: NP-Hard Reduction.

CLIQUE $\leq_P$ VC

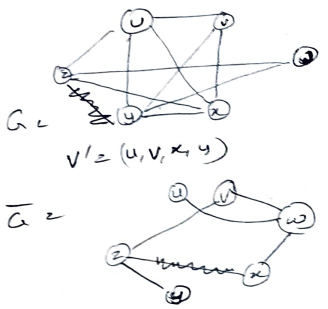
[G has
Strategy: Based on reduction of a complement of
a graph
$G = (V, E)$

$\overline{G} = (V, \overline{E})$ $\overline{E} = (u,v) \in E'$ if $(u,v) \notin E$] Polytime

G has a clique of size $|V'| = k$ iff $\overline{G}$ has a VC of $|V - V'| = |V| - k$

— Suppose G has a clique $V' \subseteq V$ with $|V'| = k$, then $(V - V') = VC(\overline{G})$
— Claim that $V - V'$ is a VC in $\overline{G}$
— Let $(u,v) \in \overline{E}$, then $(u,v) \notin E$ implies that atleast one of the ~~edge~~ u or v does not belong to $V' = VC(G)$ not $\overline{G}$
— Equivalently, atleast one of $u, v \in V - V'$ which mean the edge (u,v) is covered by $(V - V')$.
— Since (u,v) was arbitrary, every edge of $\overline{E}$ is covered by $(V - V')$
— Hence $V - V'$, $|V - V'| = k$ is a VC of $\overline{G}$.

↑ Converse



— Converse
— Suppose that $\overline{G}$ has a vertex cover $(V - (\overline{V'}))$ $V' \subseteq V$, where $|V'| = |V| - k$
— Let for all $u, v \in V$, if $(u,v) \in \overline{E}$
then $u \in V'$ or $v \in V'$ or both
— Contrapositive, for all $u, v \in V$ if $u \notin V'$ and $v \in V'$
then $(u,v) \in E$

Ham Cycle is NPC (undirected)

- A simple cycle that contains each vertex in V.
- Only a Decision problem not an optimization problem

HAM cycle $\leq_p$ NPC

$VC \leq_p$ ~~NPC~~ HAMcycle

Give an undirected graph $G = (V, E)$ and integer k.

we construct $G' = (V', E')$ having a HAM CYCLE IFF G has a VC of size k

CIRSAT
↓
3SAT
↙ ↘
IS → CLIQUE
↓
VC → SUBSET SUM
↓
HAMCYCLE
↓
TSP

CLIQUE COVER

problem

$G = (V, E)$

$\bigcup_i V_i = V$

and each V_i is a clique of G.

3-color probl

3Color $\leq_p$ CLIQUECOVER

$TSP = \{\langle G, c, k\rangle : G = (V, E)$ is complete
c is a fn from $V \times V \to Z$
$k \in Z$
G has a TS tour with cost at most $k\}$

TSP is NPComplet

Prove: Given a certificate (the sequence of n vertices in the tour, with cost of path $\leq k$), we can verify in poly time so NP.

NP Hard.

HAMCYCLE $\leq_p$ TSP

CIR
↓
3SAT
↙ ↘
CLIQ
↓
VC

- $G = (V, E)$ be an I of HAMCYCLE
- Construct I of TSP as follows.
  - $G' = (V, E')$ — complete graph
    $E' = \{(i,j) : i, j \in V \ \ i \neq j\}$
  - $c(i,j) = \begin{cases} 0 & (i,j) \in E \\ 1 & (i,j) \notin E \end{cases}$

G is undirected so no self loops, Thus $c(v,v) = 1 \ \forall v \in V$

The Instn of TSP is the $\langle G', c, 0\rangle$

- We show that:
  G has a HAMCYCLE IFF G' has a TS tour of cost at most 0
- Suppose G has a HAMCYCLE (h), then $e \in h \in E$, thus has cost 0 in G'. Thus h is a TS tour in G' with cost 0.
- Since cost of edges in E' are 0 and 1, cost of tour h' is exactly 0 & each edge on the tour must have cost 0.
  ∴ h' contains only edges in E. Conclude h' is a HAM cycle in G.

SUBSET SUM

Given S, a set of n elements & t, $t \in \mathbb{Z}$

Find $S' \subseteq S \mid \sum_{a_i \in S'} a_i = t$

- DP soln takes $O(n \cdot t)$ time
- Looks poly but is not so
  it depends on the value of the input not the number of inputs.
  i) Running time is measured as a function of input size (number of bits)
  ii) Inputs must be encoding in a reasonable succinct manner.
- Assume a_i & t are all b-bit numbers.
  rep in base 2
- Then input size is $O(nb)$
- t may be as large as 2^b
- So resulting algo has running time $O(n \cdot 2^b)$. Poly in n but exp in b

Subset SUM is a simplified version of 0-1 Knapsack problem

where value v_i = weight a_i

If we know that SUBSET SUM is NPC then we can we do for 0-1 Knapsack prob!

SUBSETSUM is NPC

Proof: Verification with indices of array whose values sum to t as certificate

NP proof

NP Hard

$VC \leq_p$ SUBSETSUM

In VC, we select vertices.

In SS, we select numbers.

So naturally we must map vertices to integers.

Constraint on all edge cover ⟹ mapped

Sum of int = target ⟹

Consider the matrix $[e_i \in E \; v_i \in V$

	e_1	e_2				e_j	
v_1	1	0	1				1
v_2		1	1				
		1	0	1	1		
	1						
v_n				1	1	1	

v_2, v_3 VC

(OR) t 1 1 1 1 1 1 for (v_2 v_3 v_7)

If t 1's Then v_2 v_3, v_7 for a VC

Problem with this approx

— logical OR ≠ addition

— No way of controlling how many 1's go into cover. (we need take all subset of 1's)

Soln

— Use base 3 or above (because each col can have max 2 1's

— Use another col with all 1's for vertices so that only 'k' can be chosen

— $R \geq 4$, then carries can occur.

- Because of logical OR & addition diff, any edge covered either one or twice can relay the sum to be a number consisting of digits 1 & 2. 1211 --- 112
This does not provide a unique target t.
- So to fix this, we create 'E' additional SLACK vars $\forall i \; 1 \le i \le E$.
for ith slack value counts of all 0's except for a single 1-digit in the ith position
Now our target would be
$t = 2222 \cdots 2$
- Now, we must understand an SLACK value can't upgrade a $0 \to 2$, but $1 \to 2$ only.
-

Reduction for $G = (V, E)$, $k \to$ t sum SS.

		e_1	e_2	...	e_E
Vertex value	v_1	1	0	1	...
	v_2	1	0	1	...
			1	0	...
			0	0	...
	v_{n-1}	1	1	0	...
	v_n	0	1	0	...
Slack var	y_1	0	1	0	0
	y_2	0	0	1	0
				0	1
				0	0
	y_E	0	0	0	0
t		k	2	2	... 2

in base 4
(Vertex cover)

let $t_4 = k222\cdots2$ be our target

$TC = O(E^2)$

We choose let's say.
$v_2\ v_3\ v_4$
$y_1\ y_2\ y_4\ y_5\ y_6\ y_7$

$4^n < t \le 4^E$
$n = |V|$

The algo will take $O(n4^n)$ time

APPROXIMATION

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n)$$

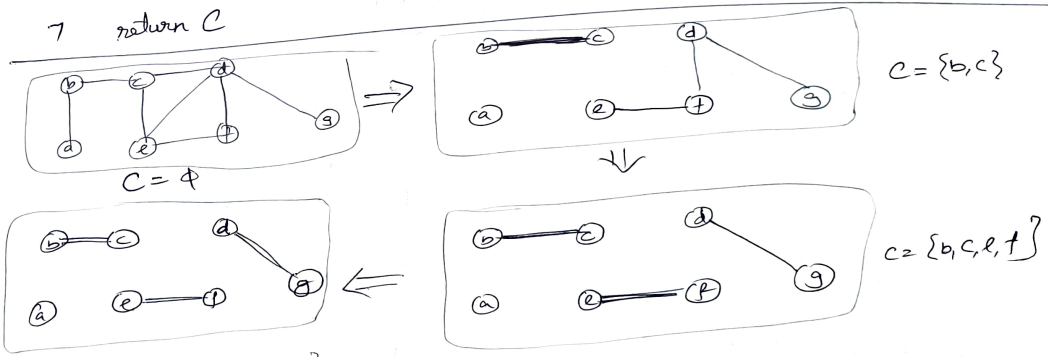
$$\rho(n) \geq 1$$

- Approximation scheme takes as input I: the instance of a problem $\epsilon > 0$ so that the scheme is a $(1+\epsilon)$ approx algo
- PTAS – if for fixed $\epsilon > 0$ the scheme is poly in n
- FPTAS – running time is poly in both $1/\epsilon$ & n
- $O\left(\left(\frac{1}{\epsilon}\right)^2 n^3\right)$ is FPTAS

$O(n^{1/\epsilon})$, $O(2^{(1/\epsilon)}n)$ are not

VC (TC = $O(V+E)$ using adjacency list to rep E')

```
1  C ← φ
2  E' ← E
3  while E' ≠ φ
4      do let (u,v) be an arbitrary edge of E'
5         C ← C ∪ {u,v}
6         remove from E' every edge incident on either u or v
7  return C
```



THM 35.1

Approx VC is a poly time-2-approx algo

Proof: Let A = set of edges picked in line 4 of approx algo

For any edge in A, C^* must include atleast one endpoint of the edge

& No 2 edges share an endpoint in A (since all others are deleted)

Thus no two edges in A are covered by same vertex from C^* and we have the lower bound

$$|C^*| \geq |A|$$

Since we pick an edge whose neither of its endpt are already in C.

$$|C| = 2|A|$$

$$|C| = 2|A| \qquad |C^*| \geq \frac{|C|}{2}$$

$$\frac{|C|}{2} = |A|$$

$$2|C^*| \geq |C|$$

~~TSP approx~~

Greedy heuristics of VC approx

— Use the ~~edge~~ vertex with high degree (instead of edge)

— Select the vertex with max degree

- Although it seems better
- It involves sorting complexity for matrices.
- No better than 2-approx.

```
GREEDYVC (G = (V, E)){
    C ← ϕ
    while (E ≠ ϕ)
        do {
            let u ∈ V with max degree
            C = C ∪ {u}
            remove all edges incident on u
        }
    return C;
```

TSP approx

Claim: Approx TSP has a ratio bound of 2

Proof: H^* → optimum tour
H → tour by our algo
T → MST

- Removing any edge of H^* makes an ~~MST~~ spanning tree and since T is MST
$$c(T) \leq c(H^*)$$
- Observe that the twice around tour has cost $2c(T)$
- By Δ inequality, when we short cut an edge of T to form H we do not increase cost of the tour, so we have.
$$c(H) \leq 2c(T)$$

~~$c(H) \leq c(H^*)$~~

Combining both.
$$\frac{c(H)}{2} \leq c(T) \leq c(H^*)$$
$$c(H) \leq 2c(H^*)$$

APPROX TSP TOUR (G, c)

1 ~~select~~ T = MST of G
$r \leftarrow$ any vertex $\in V$
$L \leftarrow$ list of vertices visited by a preorder walk of T starting with r
return L.

General TS Problem without Euclidean assumption.

~~Proof~~:

~~Let's say we have an Poly time algo~~

Claim: If $P \neq NP$, then for a ~~poly time~~ constant $\rho \geq 1$ there is no poly time algo with approx ratio ρ to the general TSP

Proof: By contradiction.

Since HAMcycle $\leq_P$ TSP

- Suppose for $\rho \geq 1$ $\exists$ a poly time approx algo A with approx ratio ρ. ρ be an integer wlog. We can use A to solve HAM cycle in poly time.
- Let $G=(V,E)$ be an instance of Ham cycle problem. We want to determine whether G is satisfiable by using the ρ factor approx algo

We turn G into an instance of a TSP problem. As follows

$G' = (V, E')$ be the complete graph on V.

$E' = \{(u,v) : u, v \in V \; u \neq v\}$

- Assign an integer cost to each edge in E'.

$$c(u,v) = \begin{cases} 1 & (u,v) \in E \\ \rho|V| + 1 & \text{otherwise} \end{cases}$$

- This ~~reduction~~ is poly time in $(|V|$ & $|E|)$
- Now if we have a soln we would use a cost of $|V|$ for a tour, if G has one.

~~Otherwise if at least one edge is not in E~~

- If G DN have a HAM cycle. any tour G' must use some edge not in E.

$$(\rho|V| + 1) + (|V| - 1) = \rho|V| + |V| > \rho|V|$$

- Thus if G has a tour, the cost is no more than $\rho|V|$, i.e ρ times cost of an optimal tour.
- Therefore we can use A to solve the Ham cycle problem in poly time.

But this is a contradiction.

SET COVER

— $X = \{x_1, x_2 \text{ —— } x_m\}$ finite set

$F = \{S_1, S_2 \text{ —— } S_n\}$ subset family

~~$x \in X$ $F \subseteq F$~~ ~~$F \neq \phi$~~

$$X = \bigcup_{S_i \in C} S_i$$

C — cover (subset cover)

$C \subseteq F$

— Optimization problem: Find min set cover.

Decision: Does $\exists$ set cover of size k

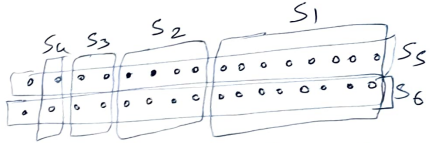
n

- Although VC has a 2-factor approx algo, SC does not
- SC has an approx bound $(\ln|n|)$ also $n = |X|$

VC as a special case of SC

$X = E$

$F = V$ and x_i can only be in two S's or only two.



Optimum $\{S_2, S_6\}$

Greedy $\{S_1, S_2, S_3, S_4, S_5, S_6\}$

GREEDY SET COVER

```
U = X   [items to be covered]
C ← φ
while (U ≠ φ){
    select S in F that covers most U
    add S to C
    U = U − S
}
return C
```

Approximation factor $(\ln(|X|))$

example

— problem with greedy it can be fooled into taking the wrong set

Showing bound does not exceed $\ln m$.

$\ln a \quad \left(1 - \frac{1}{c}\right)^c \leq \frac{1}{e}$

$c > 0$

$1 + x \leq e^x$

$1 - \frac{1}{c} \leq e^{-\frac{1}{c}}$

$\left(1 - \frac{1}{c}\right)^c \leq \frac{1}{e}$

Now proving the

Proof:

let c = optimum set cover

g = greedy cover − 1

we know that $g/c \leq \ln m$.

- Initially there are $m_0 = m$ elements left.
- We know there is a cover of size c, so by pigeonhole, there must be atleast one set that covers m_0/c elements.
- $m_1 \geq m_0 - \frac{m_0}{c}$] remaining elements left to be covered
- Similarly $m_2 \geq m_1 - \frac{m_1}{c}$
- $m_g = m_0\left(1 - \frac{1}{c}\right)^g$
- Until when?
- Until the largest value of g such that after running all but the last set of greedy cover we still have some elements left.
- Thus

$$1 \leq m\left(1 - \frac{1}{c}\right)^g$$

$$1 \leq m\left[\left[1 - \frac{1}{c}\right]^c\right]^{g/c}$$

$$1 \leq m\left(\frac{1}{e}\right)^{g/c}$$

$$\frac{1}{m} \leq \left(\frac{1}{e}\right)^{g/c}$$

$$-\ln m \leq -\frac{g}{c}$$

$$\frac{g}{c} \leq \ln m$$

$$g \leq c \ln m$$

BIN PACKING

- NP Complete
- Variant of Knapsack problem
- Given a set of n objects
  s_i – size of i^{th} object
  $0 < s_i < 1$, Bins have cap = 1 (all are same capacities)
  (Unlimited bins)
- Problem: Partition the objects into the bin so as to use the fewest possible bins.
- Simple heuristic algo
  - First fit.

First fit $\approx \frac{17}{10}$

- Find first bin that has space to hold this object
- Claim: It uses at most twice the no of bins

$$\frac{b_{ff}}{b^*} \leq 2$$

Proof: Consider an instance $\{s_1 \text{ —— } s_n\}$ of bin packing problem.

$S = \sum s_i$ denote the sum of all object sizes.

$b^* \geq S$ [since $0 < s_i \leq 1$

No of solution bins

Let $t_i \leftarrow$ total size of objects that first fit puts into bin i.

Consider bins i & $i+1$ filled by first fit. (Indexing is cyclical)

Then $t_i + t_{i+1} \geq 1$, if not then contents of both could have been put into the i^{th} bin.

$$\sum_{i=1}^{b_{ff}} t_i + t_{i+1} \geq b_{ff}$$

$$2b^* \geq 2S \geq b_{ff} \longrightarrow \frac{b_{ff}}{2} \leq b^*$$

- Best fit ($\approx \frac{17}{10}$)
- first fit decreasing [objects are first sorted]

$\approx \frac{11}{9} = 1.22$

Weighted Vertex ~~Cover~~ Vertex cover approx using Linear Programming.

$G = (V, E)$ each $v \in V$ has weight $w(v)$

For $V' \subseteq V$, $w(V') = \sum_{v \in V'} w(v)$

~~Goal~~: find min wt vertex cover.

Strategy: i) Compute a lower bound on wt of min wt VC. by using linear program

ii) Round this to obtain a VC.

but we have $x(v)$ for each $v \in V$

$x(v) \in \{0, 1\}$ → 1: in VC; 0: not in VC

Constraint for edge (u, v), atleast one of u and v must be in the VC.

as $x(u) + x(v) \geq 1$

This view → 0-1 ~~Knapsack~~ integer. prog.

ALGO:

APPROX MIN WT

$C \leftarrow \phi$

Compute $\bar{x}$, an optimal soln in RLPP

for each $v \in V$

do if $\bar{x}(v) \geq 1/2$

then $C \leftarrow C \cup \{v\}$

return C

ILP

$$\text{Min} \sum_{v \in V} w(v)\, x(v)$$

Subject to

$x(u) + x(v) \geq 1$ for each $(u, v) \in E$

$x(v) \in \{0, 1\}$ for each $v \in V$

Relaxed LPP

$$\text{Min} \sum_{v \in V} w(v)\, x(v)$$

subject to $x(u) + x(v) \geq 1$ ----

$0 \leq x(v) \leq 1$ ---

— Optimal soln here is a lower bound for ILP above.

Proof of

Claim: Algo Approx Min Wt VC is a poly time 2 approx algo to min wt VC prob (in RLPP)

Proof:
- Algo is poly time
- Let C^* be optimal soln
- z^* be value of optimal soln to the linear program.

$$z^* \leq w(C^*)$$

(include fractional sols) ↓ lower bound

- Next, we claim that by rounding the fractional values of the variables $\bar{x}(v)$ we produce a set C, → VC and satisfies:

$$w(C) \leq 2z^*$$

- Consider $(u,v) \in E$

$$x(u) + x(v) \geq 1$$

implies atleast one of $\bar{x}(u)$ or $\bar{x}(v) \geq 1/2$

Therefore atleast one of u, v will be included in C

$$z^* = \sum_{v \in V} w(v)\bar{x}(v)$$

$$\geq \sum_{\substack{v \in V \\ \bar{x}(v) \geq 1/2}} w(v)\bar{x}(v)$$] filters out $\bar{x}(v) \leq 1/2$

$$\geq \sum_{\substack{v \in V \\ \bar{x}(v) \geq 1/2}} w(v) \cdot \frac{1}{2}$$

$$= \sum_{v \in C} w(v)\frac{1}{2}$$

$$= \frac{1}{2} \sum_{v \in C} w(v)$$

$$z^* \geq \frac{1}{2} w(C)$$

$$w(C) \leq 2z^* \leq 2w(C^*)$$

SUBSET SUM problem approximation

Exponential time algo

⇓

and convert it to FPTAS

Running poly in $(\frac{1}{\epsilon})$ & (n)

ALGO.

```
1  n ← |S|
2  L0 ← <0>
3  for i ← 1 to n
4      do Li ← MergeLists(Li-1, Li-1 + xi)
5      remove from Li every ε > t
6
   return the largest elmt in Ln
```

Since the length of L_i can be as much as 2^i.

EXACT SUBSET SUM is an EXPONE-TIME ALGO

— EXACT SUBSET SUM

$S = \{x_1 \text{——} x_n\}$

$t \in \mathbb{Z}$

Iteratively compute L_i, the list of sums of all subsets of $\{x_1 \text{——} x_i\}$ not exceeding t and then it returns the max value in L_n.

$L_{i-1} + x = \{x_i + x \mid x_i \in L_i\}$

— FPTAS for SUBSET SUM,

— Truncate L_r by δ leaving as many ϵs from L_r such for each y removed from L, L' still has a z $\frac{y}{1+\delta} \le z \le y$

— Trimming takes $\Theta(n)$ time, given L is sorted in monotonically increasing order.

TRIM (L, δ)

```
m ← |L|
L' ← <y1>
last ← y1
for i ← 2 to m
    do if yi > last·(1+δ)
       then append yi onto the end
            of L'
            last ← yi
return L'
```

δ = 0.1

L = <10, 11, 12, 15, 20, 21, 22, 23, 24, 29>

L' = <10, 12, 15, 20, 23, 29.

11 rep by 10 so removed

21, 22 removed → 20

24 → 23 so 24 removed

APPROX SUBSET SUM (S, t, ϵ)

$S = \{x_1\ x_2 \text{ ——— } x_n\}$ in arbitrary order

t

ϵ approximation parameter.

$0 < \epsilon < 1$

```
1  n ← |S|
2  L0 ← <0>
3  for i ← 1 to n
4      do Li ← MERGELIST(Li-1, Li-1 + xi)
5         Li ← TRIM(Li, ε/2n)
6         remove from Li ε's > t
   let z* be largest value in Ln
   return z*
```

$S = \langle 104, 102, 201, 101 \rangle$ $t = 308$

$\epsilon = 0.40$

$\delta = \epsilon/2n = 0.40/8 = 0.05$

$L_0 = \langle 0 \rangle$

line 4: $L_1 = \langle 0, 104 \rangle$

line 5: $L_1 = \langle 0, 104 \rangle$

line 6: $L_1 = \langle 0, 104 \rangle$

$L_2 = \langle 0, 102, 104, 206 \rangle$

$L_2 = \langle 0, 102, 206 \rangle$

$L_2 = \langle 0, 102, 206 \rangle$

$\frac{104}{1+0.05} \not\leq 102$

$\frac{104}{1+0.05} \leq 102 \leq 104$

$\frac{206}{1.05} > 102$

$L_3 = \langle 0, 102, 201, 206, 303, 407 \rangle$

$L_3 = \langle 0, 102, 201, 303, 407 \rangle$

$L_3 = \langle 0, 102, 201, 303 \rangle$

$L_4 = \langle 0, 101, 102, 201, 203, 302, 303, 404 \rangle$

$= \langle 0, 101, 201, 302, 404 \rangle$

$= \langle 0, 101, 201, 302 \rangle$

returns $z^* = 302$ which is well within $\epsilon = 40\%$ of the optimal answer $307 = 104 + 102 + 101$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

$ae+bg \quad af+bh$

$ce+dg \quad cf+dh$

S_1 (B−D)
S_2 (A+D)
S_3 (A−C)
S_4 (A+B)
S_5 A
S_6 D
S_7 (C+D)

(G+H)
(E+H)
(E+F)
H
(F−H)
(G−E)
E

$S_1 = [\ \ominus\ ][+]$ $\quad S_5 \begin{bmatrix} 0 & \ominus \end{bmatrix}$

$S_{22}[+][+]$ $\quad S_6[\ 0\ \ominus\ ]$

$S_1 + 2S_{32}[\ominus \quad +]$

$S_4[+ \quad 0]$ $\quad S_4 + S_7 \begin{bmatrix} 0 \\ + \end{bmatrix}$

1 2 −4 6 $\quad$ 4 5

6 7 $\quad$ 2 −3 +5 −7

A+B
A+D
A−C
B−D
C+D
A
D

E+F
E+H
G−E
F−H
H+G
E
H

++2 $\quad$ +1
+−4 $\quad$ −3
++6 $\quad$ ++5
+−7

1 2 −4 6 $\quad$ 4 5
6 7 $\quad$ 2 −3 5 −7

DP & FLOYD WARSHALL | JOHNSON

1) DP based on M x Mul $= O(V^3 \lg V)$
2) DP (Floyd Warshall) $= O(V^3)$
3) Johnson (using adjacency list) $= O(V^2 \lg V + VE)$ time

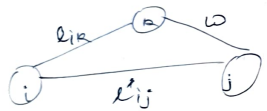
1) Show ALL pair S P problem (W) S

- This takes $O(n^4)$ time
- It will also (+ve)

$L^{(1)} = W$
$L^{(2)} = W \cdot W$
$L^{(4)} = W^2 \cdot W^2$
;

```
n ← rows[W]
L(1) ← W
for m ← 2 to n-1
    do L(m) ← Extend SHP (L(m-1), W)
return L(n-1)
```

1) DP b/w M x Mul



$O(n^3)$

Extend Shortest path (L, W)

```
n ← rows[L]
let L' = (l'ij) be a n x n Mx
for i ← 1 to n
    for j ← 1 to n
        do l'ij ← ∞
           for k ← 1 to n
               do l'ij ← min(l'ij, lik + wkj)
return L'
```

$O(n^3)$

$L^{(2^{\lceil \lg(n-1) \rceil})} = W^{2^{\lceil \lg(n-1) \rceil}}$

$2^{\lceil \lg(n-1) \rceil} \geq n-1$

so final product $= L^{(n-1)}$

FASTER ALL PSSP (W)

```
1 n ← rows[W]
2 L(1) ← W
3 m ← 1
4 while m < n-1
5     do L(2m) ← EXT Short Path(L(m), L(m))
6        m ← 2m
7 return L(m)
```

Resembles to M x Mul

$C = A \cdot B$ n x n

$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$

If we substitute

$l^{m-1} \longrightarrow a$
$w \longrightarrow b$
$l^{(m)} \longrightarrow c$
$\min \longrightarrow +$
$+ \longrightarrow \cdot$

then we get $\infty \longrightarrow 0$

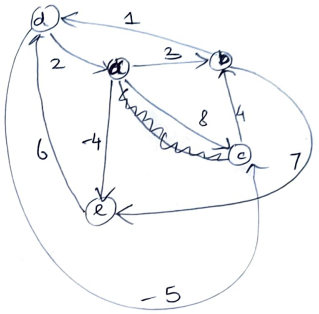
Thus it is $O(n^3)$ as in a M x M

FLOYD WARSHALL (bottoms up approach)

```
n ← nrows[W]
D^(0) ← W
for k ← 1 to n
    for i ← 1 to n
        for j ← 1 to n
            do d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
return D^(n)
```

Constructing shortest path

add the mid[i,j] = k [as a path.



a e d c b

a e d c

a e a e d

b d a 3

b d c -4

b b d 1

b d a e -1

c

Transitive closure

IF ∃ a path in a from i to j ∀ i,j ∈ V

the transitive closure of G

$G^* = (V, E^*)$ where

$E^* = \{(i,j) : \text{there is a path for } i \text{ to } j \text{ in } G\}$

Algo to find (Floyd Warshall with

- Set all ans to 1
- Initialize $t_{ij}^{(0)} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$
- For $k \geq 1$

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)} \right)$$

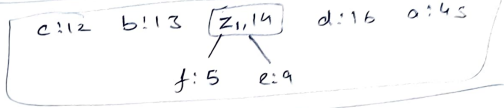
(min replaced by $\vee$, + replaced by $\wedge$)

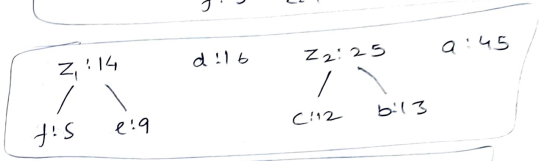
Node distance from itself $\not< 0$ means < 0 then is a negative cycle.

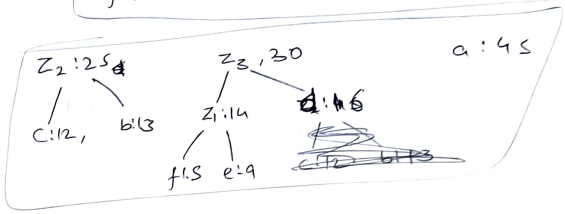
HUFFMAN

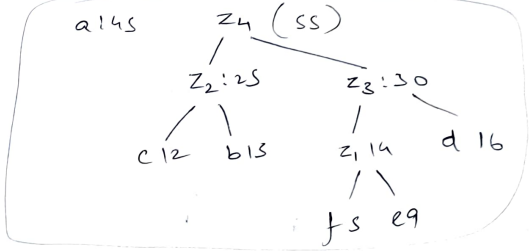
```
1  n ← |C|
2  Q ← C
3  for i ← 1 to n-1
   |  do  allocate a new node z
4  |      left(z) = Extract min(Q)
5  |      right(z) = Extract min(Q)
6  |      f(z) ← f[x] + f(y)
7  |      INSERT (Q, z)
8  |
9  return Extract min (Q)  ⟶ root of tree
```

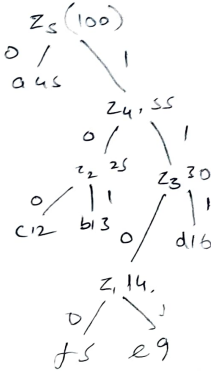
f:5, e:9, c:12, b:13, d:16, a:45











Proof of optimality

swap k & j^{th} elm $\quad l'_k = l_j \quad l'_j = l_k$

$\bar{l}' = \sum l_i p_i \quad l_j \geq l_k \quad p_j \leq p_k$

$\bar{l} = \sum l_i p_i$

$\bar{l} - \bar{l}' = l_j p_j + l_k p_k - l'_k p_k - l'_j p_j$

$l_j p_j + l_k p_k - l_j p_k - l_k p_j$

$l_j (p_j - p_k) + l_k (p_k - p_j)$

$(l_j - l_k)(p_k - p_j)$

$\geq 0 \quad \geq 0$

0-1 Knapsack Problem is FPTAS

- Given ϵ
- Compute $t = \log_{10}\left(\frac{\epsilon C_m}{n}\right)$
- Truncate t digits to each c_j i.e. $\bar{c}_j = \left\lfloor \frac{c_j}{10^t} \right\rfloor \times 10^t$
- Approx knapsack algo on $(w_1, w_2 \ldots w_n ; \bar{c}_1 \ldots \bar{c}_n ; K)$

S' → with truncated values c_i's.

Error

$$\text{error} = \sum_{j \in S} c_j - \sum_{j \in S'} c_j$$

$$\leq \sum_{j \in S} c_j - \left(\sum_{j \in S} c_j - 10^n\right)$$

$$= 10^n$$

$$\text{error} \leq 10^n$$

In general, if t digits are truncated the deviation becomes $n \cdot 10^t$

Time Complexity

$C_m = \max(c_j)$

$TC = O(n^2 C) \rightarrow \sum$ iterations

$= O(n^3 C_m)$

After truncation of t digits, the bound becomes $O(n^3 C_m 10^{-t})$

now $C_m' \leq \frac{C_m}{10^t}$

$$\frac{\sum_{j \in S} c_j - \sum_{j \in S'} c_j}{\sum_{j \in S} c_j} \leq \frac{n \times 10^t}{C_m} = \epsilon$$

$$TC = O(n^3 C_m \times 10^{-t})$$

$$= O\left(n^4 \frac{C_m}{n \cdot 10^t}\right)$$

$$O\left(n^4 \frac{1}{\epsilon}\right)$$

when $t = \left\lfloor \log_{10}\left(\frac{\epsilon C_m}{n}\right) \right\rfloor$

For ϵ error, truncate $t = \left\lfloor \log_{10}\left(\frac{\epsilon C_m}{n}\right) \right\rfloor$ digits.

Polynomial rep (Horners rule)

— $A(x) = \sum_{j=0}^{n-1} a_j x^j$

— Horners rule

$A(x_0) = a_0 + x_0(a_1 + x_0$

Point wise

$\{(x_0, y_0), (x_1, y_1), (x_2, y_2) \ldots (x_{n-1}, y_{n-1})\}$

A B 1 3
C D 5 7

E F 8 4
G H 6 2

$-32 = (3-7)(6+2)$

$S_1 = (B-D)(G+H)$

$80 = S_2 = (A+D)(E+H)$

$-48 = S_3 = (A-C)(E+F)$

$8 = S_4 = (A+B)H$

$2 = S_5 = A(F-H)$

$-14 = S_6 = D(G-E)$

$96 = S_7 = (C+D)(E)$

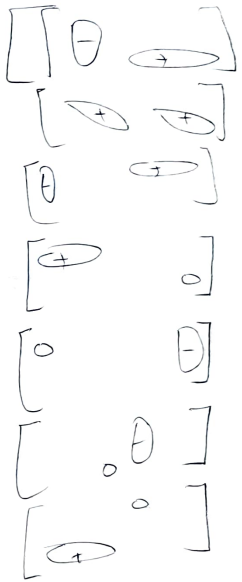
$S_1 + S_2 - S_4 + S_6$ $\quad S_4 + S_5$

$S_6 + S_7$ $\quad S_2 - S_3 + S_5 - S_7$

$-32 + 80$

$\begin{bmatrix} 26 & 10 \\ 82 & 34 \end{bmatrix}$

C D G H

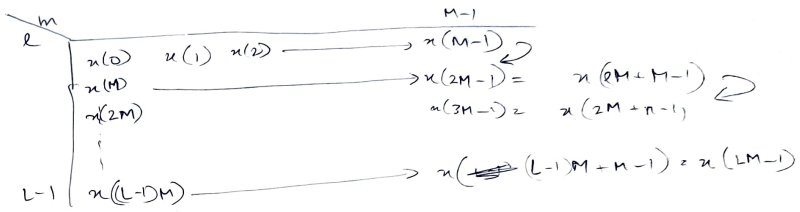


$N = 15 \quad L = 5 \quad M = 3$ DFT Calc | Two other ways

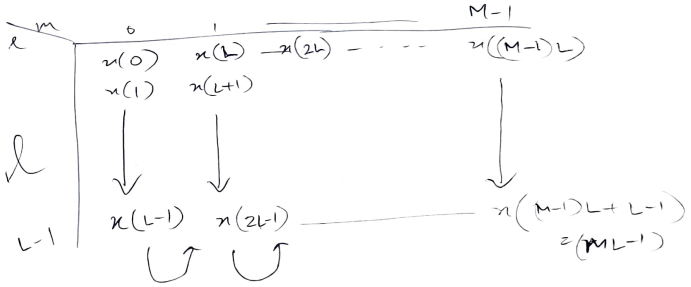
$$X(p,q) = \sum_{l=0}^{L-1} \left\{ W_N^{lq} \left[\sum_{m=0}^{M-1} x(l,m) W_M^{mq} \right] \right\} W_L^{lp}$$

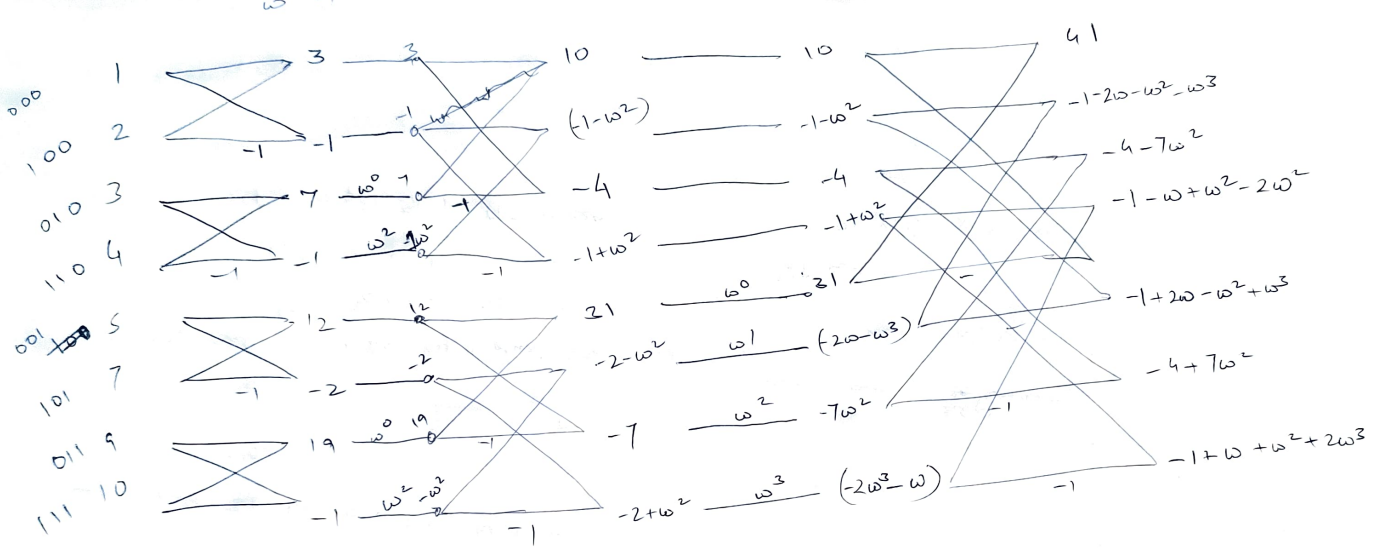
$$\Rightarrow X(p,q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(l,m) W_N^{(Mp+q)(mL+l)}$$

$n = Ml + m$



$n = Lm + l$




000 1
100 2
010 3
110 4
001 5
101 7
011 9
111 10
3
-1
7
-1
12
-2
19
-1
10
(1-ω²)
-4
-1+ω²
21
-2-ω²
-7
-2+ω²
10
-1-ω²
-4
-1+ω²
31
ω
(2ω-ω³)
ω²
-7ω²
ω³
(-2ω³-ω)
41
-1-2ω-ω²-ω³
-4-7ω²
-1-ω+ω²-2ω²
-1+2ω-ω²+ω³
-4+7ω²
-1+ω+ω²+2ω³

1 5 3 9 2 7 4 10

Using the $n = l + mL$ form [column wise progression]

$x(0,0)$ $x(0,1)$ $x(0,2)$ $x(0,3)$ $x(0,4)$

$x(1) = x(0,0)$

$x(4,0)$ $x(4,1)$ $x(4,2)$

— $N = 15$ $M = 3$ $L = 5$

3 pt DFT

$$\begin{bmatrix} f(0,0) & f(0,1) & f(0,2) \\ & f(5) & f(10) \\ \vdots & & \\ f(4,0) & f(4,1) & f(4,2) \end{bmatrix}$$

— Compute 3 pt DFT for each of the 5 rows

$G(0,0)$ $G(0,1)$ $G(0,2)$

$G(4,0)$ ——————

— Multiply each term $f(l,q)$ by a phase factor

$W_N^{lq} = W_{15}^{lq}$ $0 \le l \le 4$ $0 \le q \le 2$

Result = G MX ——————

— Final step: Compute 5 pt DFT for each column.

Result is read row wise.

$X(0)$ $X(1)$ $X(2)$

$X(3)$ $X(4)$ — $X(5)$

$X(12)$ $X(13)$ $X(14)$

Col wise

1) x given in column wise (store signal column wise)

↓

2) Compute #Col - pt DFT

↓

3) multiply by phase factors W_N^{lq} $0 \le q \le$ #Col $0 \le l \le$ #Rows

4) Compute #Rows - pt DFT

Given sequence row wise.

$x(1)$ $x(5)$ $x(10)$ $x(2)$ $x(6)$ $x(11)$ --- $x(4)$ $x(9)$ $x(14)$

O/P seq.

$X(0)$ $X(1)$ $X(2)$ $X(3)$ —————— $X(14)$

Row wise storage of x

i) Store signal row wise

$$\begin{bmatrix} x(1) & x(2) & x(3) & x(M-1) \\ x(M) & x(M+1) & \cdots & \vdots \\ x(2M) & & & \vdots \\ \vdots & & & \vdots \\ x((L-1)M) & \cdots & & x(ML-1) \end{bmatrix}$$

ii) Compute column wise DFT [L pt DFT for each column]

iii) Multiply by phase factors

iv) Compute row wise DFT [M point to each row]

v) Read result column wise.

When N is highly composite then

$N = n_1 n_2 \cdots n_p$ (n_i are prime)

In case $n_1 = n_2 = \cdots = n_p = 2$, we use a radix-2 FFT

Radix-2 FFT Algo

$n = 2$

$N = 2^k$

2^2

0, 1, 2, 3

0 2
1 3

0 2 $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
1 3

$= \begin{bmatrix} 2 & 4 \\ -2 & 2 \end{bmatrix}$ → M

$\begin{bmatrix} 2 & -2 \\ 4 & -2 \end{bmatrix}$ ↓ L

Calc

Mult

DFT (0,2), DFT(1,3)

w_4^{lq} $0 \le l \le L-1$

$0 \le q \le M-1$

$\begin{matrix} 2 & -2 \\ 4 & -2 \end{matrix}$ $\begin{bmatrix} 2 & -2 \\ 4 & -2j \end{bmatrix}$

0 1 2 3 | 1 1 1 1 | 6
1 j -1 -j | -2-2j
1 -1 1 -1 | -2
1 -j -1 j | -2+2j

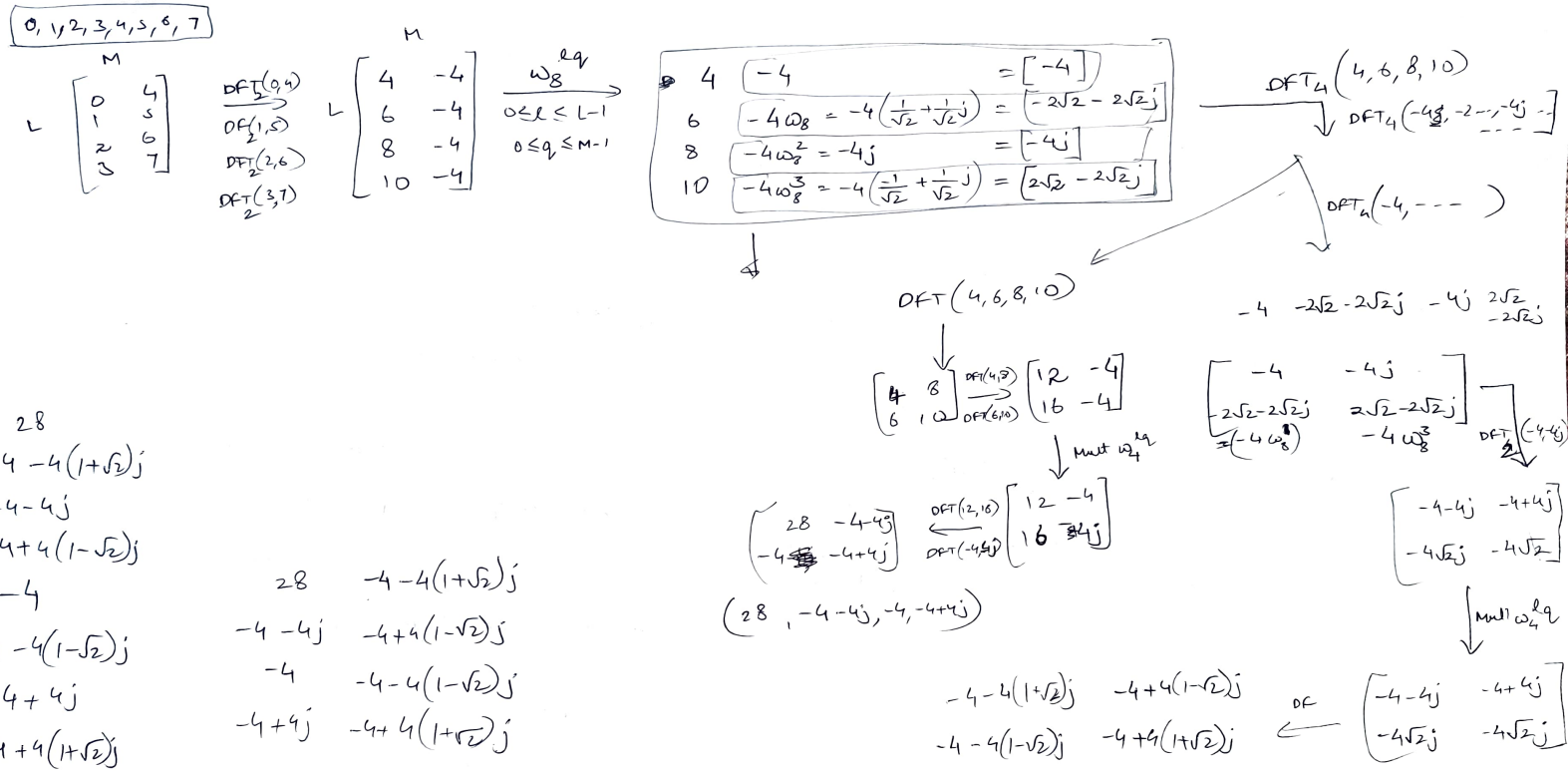
Calc

DFT $\begin{bmatrix} 2 \\ 4 \end{bmatrix}$, DFT $\begin{bmatrix} -2 \\ -2j \end{bmatrix}$

6 $\quad -2-2j$

$-2 \quad -2+2j$

$[6, -2-2j, -2, -2+2j]$


0, 1, 2, 3, 4, 5, 6, 7
M
L
DFT2(0,4)
DFT2(1,5)
DFT2(2,6)
DFT2(3,7)
M
L
ω_8^{lq}
$0 \le l \le L-1$
$0 \le q \le M-1$
4 | -4 = [-4]
6 | $-4\omega_8 = -4\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}j\right)$ = $[-2\sqrt{2} - 2\sqrt{2}j]$
8 | $-4\omega_8^2 = -4j$ = [-4j]
10 | $-4\omega_8^3 = -4\left(\frac{-1}{\sqrt{2}} + \frac{1}{\sqrt{2}}j\right)$ = $[2\sqrt{2} - 2\sqrt{2}j]$
DFT4(4, 6, 8, 10)
DFT4(-4, -2-2j, -4j, ---)
DFT4(-4, ---)
DFT(4, 6, 8, 10)
-4 -2√2-2√2j -4j 2√2-2√2j
DFT(4,8)
DFT(6,10)
Mult ω_4^{lq}
DFT(2,10)
DFT(-4,4j)
28 -4-4j -4 -4+4j
28 , -4-4j, -4, -4+4j
DFT
Mult ω_4^{lq}
28
-4 -4(1+√2)j
-4-4j
-4+4(1-√2)j
-4
-4-4(1-√2)j
-4+4j
-4+4(1+√2)j
28 -4-4(1+√2)j
-4-4j -4+4(1-√2)j
-4 -4-4(1-√2)j
-4+4j -4+4(1+√2)j

Chap 5. Probabilistic Analysis & Randomized Algo.

The Hire Assistant problem

```
HIRE-ASSISTANT(n)
1   best ← 0
2   for i ← 1 to n
3       do interview candidate i
4          if candidate i is better than candidate best
5             then best ← i
6                  hire candidate i
```

- Running time of algo not a concern, cost of interviewing & hiring is.
- If c_i — cost of interviews
  c_h — hiring cost
  m — # hires
  then total cost = $O(nc_i + mc_h)$
  - nc_i → always incurred
  - mc_h → hires

- worst case of mc_h → hire all candidates in increasing order of quality
- What about avg case — probabilistic analysis
- Probabilistic Analysis
  - use knowledge / make assumption about the input distribution
- Randomized algo.
  - Enforced random order.
  - An algo is randomized if
    - its behavior is determined not only by its input but also by values produced by a random number generator.
    - RANDOM(a,b) function
      randomly b/w [a, b] uniformly

After randomization.

1 —— (i-1) i

↑

This candidate has:

$\frac{1}{i}$ prob of being selected in line 5.

$X_i = I(\text{cand } i \text{ is hired}) = \begin{cases} 1 & \text{if hired} \\ 0 & \text{not hired} \end{cases}$

$E(X_i) = P_r(\text{candidate } i \text{ is hired})$

$\therefore E(X_i) = \frac{1}{i}$

$X = X_1 + X_2 \cdots X_n$

Taking on all X_i's

$$E(X) = E\left[\sum_{i=1}^{n} X_i\right]$$

$$= \sum_{1}^{n} E(X_i)$$

$$= \sum_{1}^{n} \frac{1}{i}$$

$$E(X) = \ln n + O(1)$$

approx $\ln(n)$ candidates are hired

$m \approx \ln(n)$

Thus total hiring cost $= O(c_n \ln(n))$

Hire exactly twice [Random ord

(3) (4) (2) (5) (6) (7) (8) (1) (9) (10)

$$8!\,({}^{9}C_1) + {}^{8}C_1(8!) + 8!\,({}^{7}C_1)$$

$$8!\left({}^{9}C_1 + {}^{8}C_1 + {}^{7}C_1 + {}^{6}C_1 + {}^{5}C_1 + {}^{4}C_1 + {}^{3}C_1 + {}^{2}C_1 + {}^{1}C_1\right)$$

$$\frac{}{10!}$$

$$\frac{9! \cdot 5}{10!} = \frac{1}{2}$$