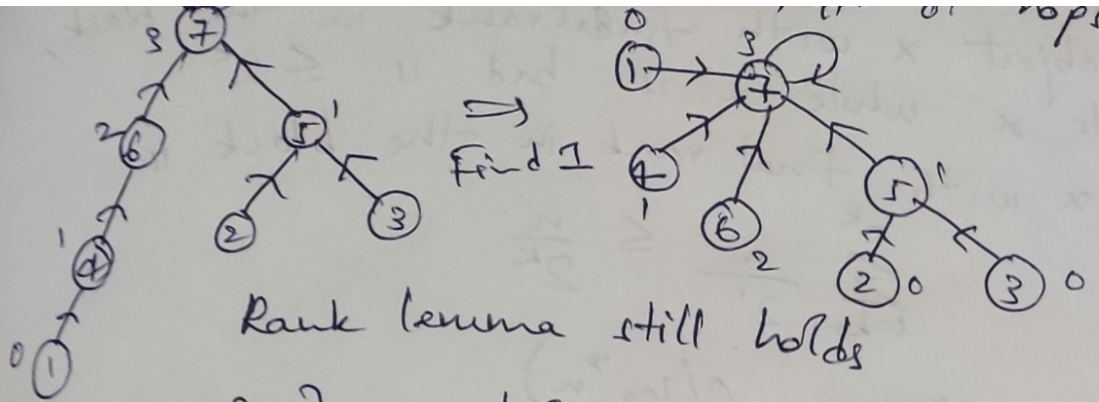


Find 1

Rank lemma still holds

rank[parent[x]] > rank[x] for all non-roots.

Thm:- With union by rank and path compression in UNION + FIND operations takes $O(m \log^* n)$ time

$\log^*(2^{65536}) = 5$

$2^{2^{2^{2^{2}}}}$ (16 twos)

Hopcraft - Ullman

9/9/23

Pf:- rank[parent[x]] − rank[x] - measure the progress

Rank blocks :- {0}, {1}, {2,3,4}, . . . $\{2^k+1 \ldots 2^{2^k}\}$

different rank blocks = $O(\log^* n)$ - - - -

Defn:- At a given pt in time, call object x good if

i) x or x's parent is a root (OR)

ii) rank[parent[x]] in larger block than rank[x].

x is bad otherwise.

Every FIND visits only $O(\log^* n)$ good nodes.
(2 + # rank blocks)

Total work done during m operations $O(m \log^* n)$
+ total # of visits to bad nodes.

Consider a rank block $\{k+1, \ldots, 2^k\}$

For each object x with final rank in the block, # visits to x while x is bad is $\leq 2^k$

objects x with final rank in the block is

$$\sum_{i=k+1}^{2^k} \frac{n}{2^i} \leq \frac{n}{2^k}$$

blocks $O(\log^* n)$

Total work $= O((m+n)\log^* n)$

Tarjan:- m find + Union operat^{n}s $O(m\alpha(n))$ (inverse Ackar f^n

Ackermann f^n:-

$k \geq 0, r \geq 1$, $A_k(r)$

$A_0(r) = r+1$

$A_k(r) = \underbrace{A_{k-1} \circ A_{k-1} \circ \cdots \circ A_{k-1}}_{r \text{ times}}(r)$

$A_1(r) = \underbrace{A_0(A_0(\cdots A_0}_{r \text{ times}}(r))$

$= A_0 A_0(r+1)$

$= 2r$

$A_2(r) = \underbrace{A_1 \circ A_1 \circ \cdots A_1}_{r \text{ times}}(r)$

$= \cdots (2(2r))$

$= 2^r r$

$A_3(r) =$

$A_3(2) = A_2(A_2(2))$

$= A_2(8)$

$= 2^8 + 8 = 204$

$A_4(2) = A_3(A_3(2))$

$= A_3(2048)$

$= A_2 \cdots A_2($

Clearly $A_k(r) \geq$

Def^n: For every $n \geq 4$, $\alpha(n) =$ min k such that $A_k(2) \geq n$

$\alpha(1) = 1$ $\quad n = 4$

$= 2$ $\quad n = 5 \ldots 8$

$= 3$ $\quad 9 \ldots 2048$

$= 4$ $\quad n$ tower of 2's of height 2048

Ackermann analysis — DSA by Kozen

n-distinct objects — UNION + FIND (lazy union, by rank, path comp)

seq m

$O(m \alpha(n))$

Def^n $\delta(x) =$ max value of $k \Rightarrow$ rank[parent[x]] $\geq A_k$[rank(x)]

$\delta(x) =$ max value of $k \Rightarrow$ rank[parent(x)] $\geq A_k$[rank(x)]

$\delta(x) \geq 0$

$\delta(x) \geq 1 \iff$ rank[parent[x]] ≥ 2 rank[x]

$\delta(x) \geq 2 \iff$ rank[parent[x]] $\geq$ rank[x] $\cdot 2^{\text{rank}[x]}$

For all objects x with rank[x] ≥ 2, $\delta(x) \leq \alpha(n)$

[since $A_{\alpha(n)}(2) \geq n$]

Good vs bad objects

Def^n: An object x is bad if all of ↓ hold.

i) x is not root

ii) parent(x) is not a root

iii) rank[x] ≥ 2

iv) x has an ancestor y with $\delta(y) = \delta(x)$

$= 2^{2^{\cdot^{\cdot^{\cdot}}}}$ height 2 of 8

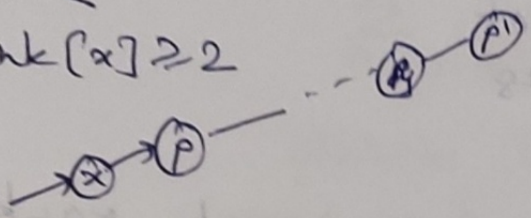
—— iterated towers (Wowzer f^m)

max-# of good objects on an object-root path
$= \Theta(\alpha(n))$

$1 + 1 + 1 + 1 + \alpha(n)$

$f(x) = k$

rank[x] ≥ 2



path compression :- x's new parent will be p' or even higher

$$\text{rank}\{x\text{'s new parent}\} \geq \text{rank}[p'] \geq A_k[\text{rank}[y]] \geq A_k[\text{rank}[p]]$$

$r = \text{rank}[x] \geq 2$

After "r" such pointer updates

$$\text{rank}[x\text{'s parent}] \geq \underbrace{A_k \circ \cdots \circ A_k}_{r \text{ times}}(r) = A_{k+1}(r)$$

While x is bad, every "r" visits increases f(x)

$\Rightarrow \leq r\alpha(n)$ visits to x while it is bad

Total # of visits to bad objects

$$\leq \sum_{\text{objects } x} \text{rank}[x]\,\alpha(n)$$

$$= \alpha(n) \sum_{r \geq 0} r\,(\#\text{ of objects with rank } r) \quad \left(\leq n/2^r\right)$$

$$= n\alpha(n) \underbrace{\sum_{r \geq 0} r/2^r}_{O(1)}$$

$$= O(n\alpha(n))$$

6/9/23

APSP :-

G - directed, c_e - edge costs

o/p :- shortest path distances of $u \rightsquigarrow v$ path $\forall$ $u, v \in V$

(or) report the presence of -ve cycle

Running time :-

n-Dijkstra (no -ve weights)

$O(nm \log n)$ — $m = O(n^2)$ — $O(n^3 \log n)$ Dense

$m = O(n)$ — $O(n^2 \log n)$ Sparse

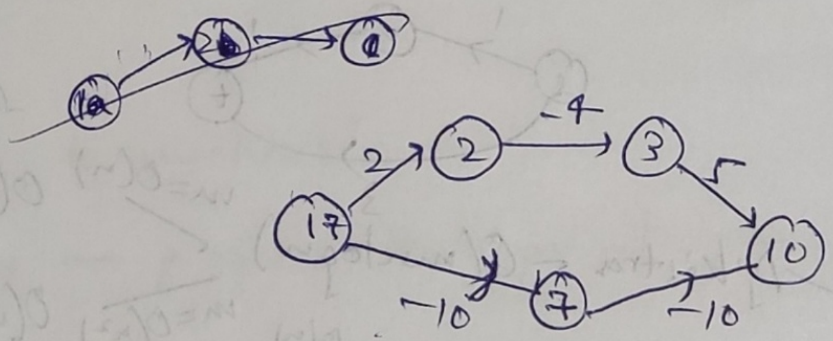
n-Bellmann Ford (general edge costs)

$O(n^2 m)$ — $m = O(n^2)$ — $O(n^4)$

$m = O(n)$ — $O(n^3)$

Lemma :- Suppose G has no -ve cycle. Fix source $i \in V$, destinat^n $j \in V$ and $k \in \{1, 2, \ldots, n\}$.

P = shortest path (cycle-free) with all internal nodes from $V^{(k)}$.



Then :-

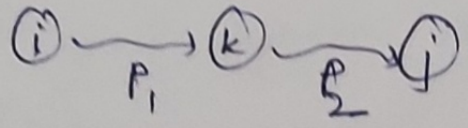
Case 1 :- If k is not internal to P, then P is a s.p $i \rightsquigarrow j$ with all internal vertices in $V^{(k-1)}$.

Case 2 :- If k is internal to P

P_1 = s.p $i \rightsquigarrow k$ from $V^{(k-1)}$

P_2 = s.p $k \rightsquigarrow j$ " "

RE for Lemma :- Exercise



$B[i,j,0] = 0 \quad \text{if } i=j$

$= c_{ij} \quad (i,j) \in E$

$= \infty \quad (i,j) \notin E$

For k=1 to n

For i=1 to n

For j=1 to n

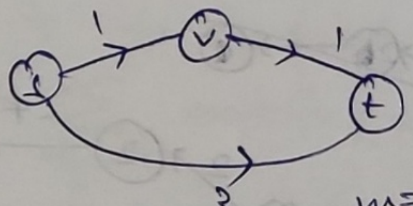
$$B[i,j,k] = \min\{B[i,j,k-1], B[i,k,k-1] + B[k,j,k-1]\}$$

$B[i,i,n] < 0$ for at least one $i \in V$ at the end - report -ve cycle

Johnson's Algo

$O(mn\log n) + O(mn)$

G - directed graph with general edge lengths



↳ #v → t is

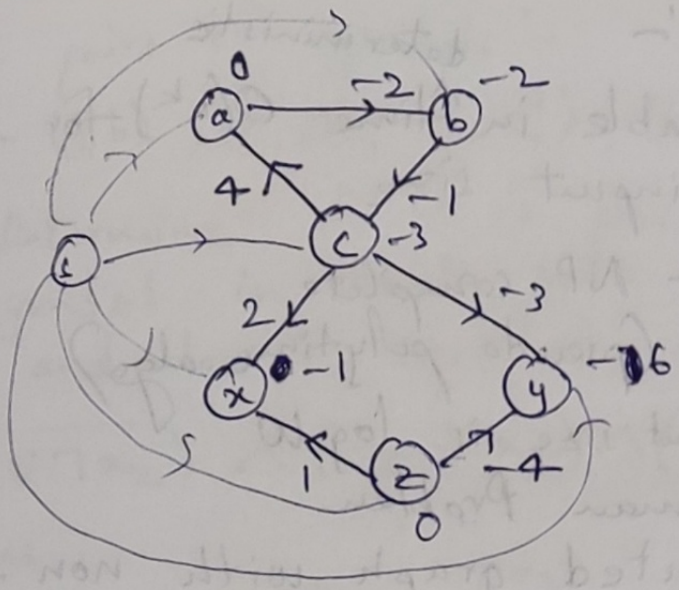
10/4/23

APSP :-

- Dijkstra - $O(mn\log n)$
  - $m = O(n)$ → $O(n^2 \log n)$
  - $m = O(n^2)$ → $O(n^3 \log n)$
- BF - $O(nmn)$
  - $m = O(n)$ → $O(n^3)$
  - $m = O(n^2)$ → $O(n^4)$

General case

$O(n^3)$ - Floyd Warshall



→ — 0 weights

$G + \{s\} = G'$

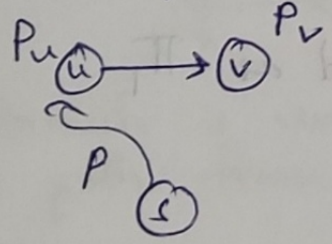
- Run BF on G' with "s" as source
- Run Dijkstra's for n times
- return $d(u,v) = d'(u,v) - p_u + p_v$

$O(n) + O(nm) + O(m) + O(mn \log n) + O(n^2)$
$= O(nm \log n)$

Assigning source vertex, Dijkstra

For every edge $e = (u,v)$ of G, is non-ve

T.P $c_e' = c_e + p_u - p_v$



N-P completeness

Halting problem is unsolvable

non exp:-

Quines — programe that print themselves

Solvable:-

Intractable:-

└ only have brute force algo in exp time

Polynomial Solvability:- deterministic

P - all problems solvable in time $O(n^k)$ for some k where n - input size

Knapsack - $O(nW)$ - NP complete

(Pseudo polytime algo)

input size is $\log W$

14/9/23

TSP:- Travelling Salesman Problem

i/p: Complete undirected graph with non-ve edge costs.

Qn:- A min-cost tour?

Conjecture:- There is no poly-time algo for TSP

Reduct^n:- a Problem Π_1 reduces to Π_2 if given a poly-time subroutine to Π_2 can be used to solve Π_1.

Completeness:-

- Suppose Π_1 reduces to Π_2

Π_2 is atleast as hard as Π_1

Def^n:- $\mathcal{C}$ = a set of problems

problem Π is $\mathcal{C}$-complete

if i) $\Pi \in \mathcal{C}$

ii) everything in $\mathcal{C}$ reduces to Π.

Class NP:-

⇓

Non deterministic polytime algo

If i) Solutions always have length polynomial in the input size

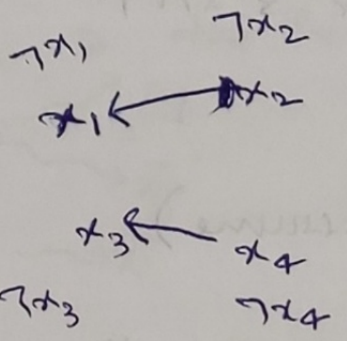
ii) Proposed solutions can be verified in poly-time

Set of pairs of symbols :- 2 SAT Problem $\in P$
ex :- (A, b), (E, D), (d, C), (b, a) 3 SAT is NP-complete.

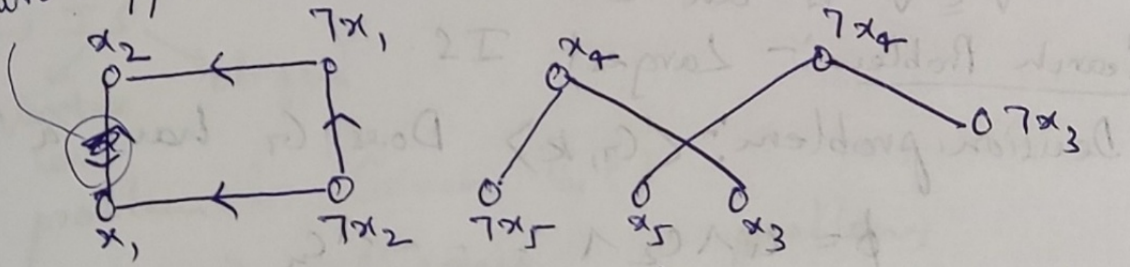
Qn :- Determine if it is possible to circle atleast one symbol in each pair without circling upper and lower case of same symbol.

$(x_1 \vee \neg x_2) \wedge (x_5 \vee x_4) \wedge (\neg x_4 \vee x_3) \wedge (\neg x_2 \vee \neg x_1)$

clauses



Sir wrote opposite



Claim :- formula is unsatisfiable $\Leftrightarrow$ there is some x with a path from $x \rightsquigarrow \neg x$ & $\neg x \rightsquigarrow x$.

Pf :- $\Leftarrow$

$x \Rightarrow \neg x$ & $\neg x \Rightarrow x$

$\Rightarrow$ There is no x s.t. $x \rightsquigarrow \neg x$ ~~or~~ or $\neg x \rightsquigarrow x$

$x = T$

- Assign T to all variable ~~reducable~~ reachable from x
- Assign negat^n of those variables false

While this algo identifies $p \to q$ for 2 SAT, it identifies $p \to q \vee r$ for 3 SAT.

Search vs Decision

3 SAT is NP-Complete (Assume)

Ind-Set $G = (V, E)$

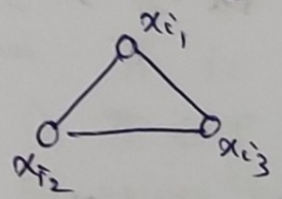
$V' \subseteq V$ is an IS if for all $u, w \in V'$, $(u, w) \notin E$

Search Problem:- Largest IS.

Decision problem:- $\langle G, k \rangle$ Does G have a IS of size $\geq k$?

$$\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$$

$$C_i = (x_{i_1} \vee x_{i_2} \vee x_{i_3})$$



Vertex-cover:- $G = (V, E)$

$V' \subseteq V$ is a vertex cover.

$(u, v) \in E$ — either $u \in V'$ or $v \in V'$

(G, k) — Does G have a vertex cover of size $\leq k$?

Soln:- V^c where V is the IS.

If $V' \subseteq V$ is an IS of size k, then $V - V'$ is a Vertex Cover of size $n-k$

Clique :- $G = (V, E)$

$V' \subseteq V$ is a clique for all $u, v \in V'$, $(u, v) \in E$

The clique of G is IS of G^T.

In Michael Sipser, Theory of Computat^n :-

21/4/23

Both $x_i \rightsquigarrow \bar{x}_i$ & $\bar{x}_i \rightsquigarrow x_i$ can't be there

→ Pick a literal x_i ∄ there is no $x_i \rightsquigarrow \bar{x}_i$.

$x_i \rightsquigarrow x_j$

$x_i \rightsquigarrow \bar{x}_j$

$x_j \rightsquigarrow \bar{x}_i$

3SAT → IS → clique

Vertex Cover

Subset sum problem :- Given (S, t)

Special instances of NP-comp... problems

1) WIS in a path / in a tree — ?

2) Heuristic :- Algorithms which are not always correct.

— Karger's algo

3) TSP :- $n! \neq n^n$

Exact algo for VC

i/p :- $G = (V, E)$ - undirected

Q^n :- Is there a VC of size $\leq k$

Star graph - 1

Complete graph K_n - $n-2$

Substructure Lemma:-

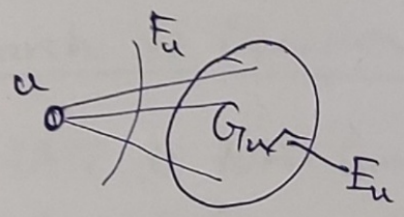
$G, (u,v) \in E, k \geq 1$

G_u = G with u & its neighbouring edges removed

G_v =

G has a VC of size $k \Leftrightarrow G_u$ (or) G_v (or) both has a VC of size $k-1$.

Pf:- ($\Leftarrow$) G_u S - VC of size $k-1$



$S \cup \{u\}$ is VC of G of size k

($\Rightarrow$) S is a VC of G of size k $u \in S$ $S \setminus \{u\}$ $(u,v) \in E$

Search Algo:-

$G = (V, E)$, k, $|V| \geq 2$, $k \geq 2$ — remaining cases are base cases

For all edges:-

1. Pick an arbitrary $(u,v) \in E$ ~~For all~~
2. Recursively search a VC of size $k-1$ in G_u ($S \cup \{u\}$)
3. " in G_v ($S \cup \{v\}$)

4) Both FAIL, No VC of size "k".

Correctness:- Induxn + substructure lemma

20/4/23

NP- completeness :-

1. Special tractable sub classes
2. Heeristic
3. Exact exponential time algo (better than brute force)

VC - $O(2^k m)$

TSP - $O(2^n n^2)$

i/p :- Complete undirected graph, non-ve edge costs

o/p :- min-cost tour.

L_{ij} = length of SP from 1 to j that uses exactly i edges and no repeated vertices

$i \in \{0, 1, \ldots, n\}$, $j \in \{1, 2, \ldots, n\}$

Recurrence

$$L_{ij} = \min_{k \neq 1, j} \{L_{ik} + c_{kj}\}$$

$S \subseteq \{1, 2, \ldots, n\}$ that contains 1 & j

$j \in \{1, 2, \ldots n\}$

L_{Sj} = min length of a path from 1 to j that visits precisely the vertices of S (exactly once)

$$L_{Sj} = \min_{\substack{k \in S \\ k \neq j}} \{L_{S-\{j\}, k} + c_{kj}\}$$ - size of subproblem $|S|$

A = 2-d array, index by subsets $S \subseteq \{1, 2 \ldots n\}$ that contains 1 & destinations $j \in \{1, \ldots, n\}$

$$A[S,i] = \begin{cases} 0 & \text{if } S = \{1\} \end{cases}$$

For $m = 2, 3, \ldots, n$ [m = subprob size]
  For each set $S \subseteq \{1,2,\ldots,n\}$ of size m that contains 1
    For each $j \in S$, $j \neq 1$

$$A[S,j] = \min_{\substack{k \in S \\ k \neq j}} \{A[S-\{j\}, k] + C_{k,j}\}$$

Return $\min_{j=2\ldots n} \{A[\{1,2,\ldots,n\}, j] + C_{j1}\}$

$O(n \cdot 2^n) \cdot O(n)$ ← work per subproblem
- n: choice of j
- 2^n: choice of S

Knapsack :-
- $O(nW)$
- greedy heuristic – $O(n \log n)$

1) Sort and render item so that

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \cdots \geq \frac{v_n}{w_n}$$

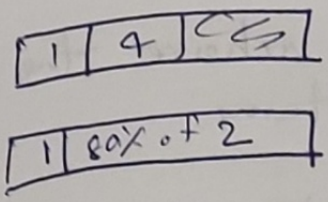
2) Pick items in this order until one doesn't fit.
3) Return either step 2 sol^n (or) the max valuable item, whichever is better. that fits.

Thm :- Value of the 3-step greedy sol^n is always $\geq 50\%$ value of optimal sol^n.

22/9/23

Thm :- Value of 3-step greedy $\geq$ 50% value of an optimal sol^n

$F \geq OPT$



Value of 3-step greedy $\geq$ total value of the 1st k items

$\geq$ Value of the $(k+1)^{th}$ item

2(Value of 3-step greedy) $\geq$ total value of 1st $(k+1)$ items

$\geq$ total value of greedy fraxnal sol^n

$\geq$ optimal knapsack problem

Q) $W = 1000$

$v_1 = 502$ $v_2 = w_2 = 500$

$w_1 = 501$ $v_3 = w_3 = 500$

Optimal :- $V = 1000$ $W = 1000$

Greedy :- $V = 502$ $W = 501$

Suppose every item i has size $w_i \leq 10\%$ of W

Value of 2 step greedy $\geq$ 90% value of an optimal sol^n

$\Downarrow$ $\Uparrow$

90% value of greedy fraxnal sol^n

$\max w_i \leq \delta W$

then 2-step greedy value $\geq (1-\delta)$ optimal

Arbitrarily good approx :-

$\epsilon > 0$ (user-specified)

Guarantee $(1-\epsilon)$- approximat^n

& Running time (vs) accuracy

$O(n^2 v_{max})$ time dp

$v_{max} = \max_i \{v_i\}$

$O(nW)$

$S_{i,X}$ = min total size needed to achieve the value $\geq X$ using the 1st "i" items.

$$S_{i,X} = \min \begin{cases} S_{i-1,X} & i \text{ is not used in the opt sol}^n \\ w_i + S_{i-1,X-v_i} \end{cases}$$

Return the largest X such that

$A[n,X] \leq W \leftarrow O(n^2 V_{max})$

1. $v_i^x = \left\lfloor \frac{v_i}{m} \right\rfloor$ for every item i

2. Compute optimal sol^n w.r.t $\hat{v}_i$ using the dynamic prog algo.

$m\hat{v}_i \in [v_i - m, v_i]$

$v_i \geq m\hat{v}_i \geq v_i - m$ (① ②)

$S^* =$ opt. sol^n to the original sol^n

$S =$ our heuristic sol^n [optimal for $\hat{v}_i$'s]

$$\sum_{i \in S} \hat{v}_i \geq \sum_{i \in S^*} \hat{v}_i \rightarrow ③$$

①
$$\sum_{i \in S} v_i \geq m \sum_{i \in S} \hat{v}_i \geq m \sum_{i \in S^*} \hat{v}_i \geq \sum_{i \in S^*} (v_i - m)$$

$$\sum_{i \in S} v_i \geq \sum_{i \in S^*} v_i - mn$$

Constraint:- $\sum_{i \in S} v_i \geq (1-\epsilon) \sum_{i \in S^*} v_i$

$$m n \leq \epsilon \sum_{i \in S^*} v_i$$

$$\boxed{m = \frac{\epsilon v_{max}}{n}}$$