Quiz in tutorial on 3rd (Lec19, 20)

CSE525 Lecture 20 M21

Knapsack(DP) = {(v1,w1)...(vn,wn), W} : Maximize value of items whose total wt is atmost W.

KNAPSACK(S={(v1,w1), ... (vn,wn)}, W, V): Is there a subset of items whose total value is at least V and total weight is at most W?

PARTITION(A={a1 ... am}) : Can the array be partitioned into two parts with equal sums?
SUBSETSUM(A={a1 ... am}, T): Is there a subset of A whose sum is T?

sum = T $\Leftrightarrow$ sum $\geq$ T & sum $\leq$ T

Detour

OPT(x): Maximize f(x)
DECOPT(x, K): Is maximum f(x) $\geq$ K?

These two are polynomially equivalent.

Small overhead allows us to solve one using the other

Given algo for OPT (AlgoOPT) $\to T_1(x)$
design algo for DECOPT?

def AlgoDECOPT(x, K) : return AlgoOPT(x) $\geq$ K ; //running time: $O(T_1(x))$

Given algo for DECOPT (AlgoDECOPT $\to T_2(x)$), design : def AlgoOPT(x):
for K = 1 ... trivial limit (Z):
if AlgoDECOPT(x,K) is false:
return K-1 ;

Z for Knapsack : $\sum v_i$ running time: $O(T_2(x) * Z)$

$|v_i| = \lg(v_i)$ $v_i = 2^{\lg(v_i)} = 2^{|v_i|} \neq poly(\lg(v_i))$

$|\sum v_i| = \sum \lg(v_i)$

running time: $O(T_2(x) * \boxed{\sum v_i})$ $\sum v_i$ is not $poly(\sum \log(v_i))$

not a polynomial time reduction

1000 number of digits = 4, number of bits = 9 $\lg_{10}(x) = \lg_2(x) * \lg_{10} 2$

x $= \lceil \lg_{10}(x) \rceil$ $= \lceil \lg_2(x) \rceil$ $= \Theta(\lg_2(x))$

$|x|$ = #digits/#bits $= O(\lg_2(x)) =$

Design a binary search: determine largest K s.t. DECKNAPSACK(x,K) return Yes.

$O(T_2(x) * \lg(Z)) = O(T_2(x) * \lg(\sum v_i))$

SUBSETSUM $\leq$ KNAPSACK: // Knapsack is a general form of SUBSETSUM

Reduce(A,T) = (S, W, V)

def Reduce (A, T) :
return {$(a_1, a_1) \cdots (a_n, a_n)$},
W = T
V = T

? works?
{$(1,a_1) \cdots (1,a_n)$}
W = T
V = n

Lemma: A = {$a_1 \cdots a_n$} has a subset with sum $\geq$ T & sum $\leq$ T iff S has some items with value $\geq$ V & wt $\leq$ W

SCHEDULE(homeworks {H1... Hn}): decide if all these homeworks can be submitted?

Each Hi = (ai : announce time, si: solving time, di: deadline) // parallel jobs not possible

Reduce from SUBSETSUM(X={x1... xn}, T): Is there a subset of X whose sum is T?

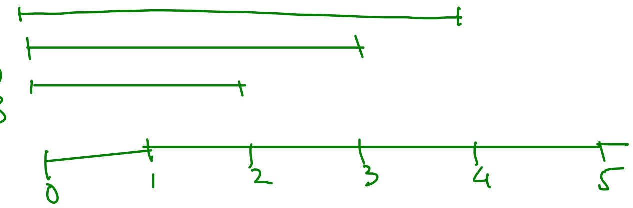
SUBSETSUM $\leq$ SCHEDULE

Lemma:(0) Reduction is polynomial time.

(i) If X has a subset Y whose sum is T then there is a way to schedule all Hi's.

(ii) If there is a way to schedule all Hi's then there exists a subset X' of X with sum T.

def Reduce($X=\{x_1 \dots x_n\}$, T): $X=\{2,3,4\}$



× return $\{H_i=(0, x_i, x_i)\}$

× return $\{H_i=(0, x_i, \underbrace{\sum x_i+1}_{\text{Sum } O(lg)})\}$ ← can be always scheduled even though X has no good subset.

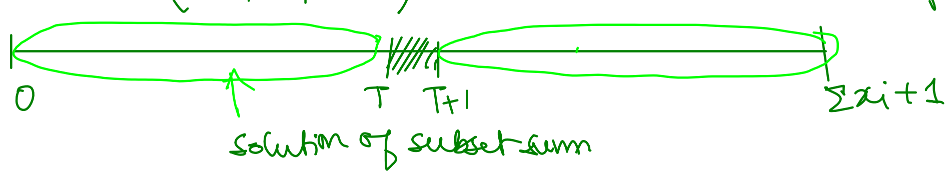
× return $\{H_i=(x_i, x_i, 2x_i)\}$

× return $\{H_i=(0, x_i, T)\}$ $X=\{2,3,4\}, T=1000$

✓ $H_i=\{(0, x_i, \sum x_i+1)\}$ for $i=1 \dots n$

n+1 homeworks

$H_{n+1} = (T, 1, T+1)$ ← must be done from T to (T+1)



(Not part of Quiz syllabus)

Reduce IS (independent set) to SUBSETSUM !

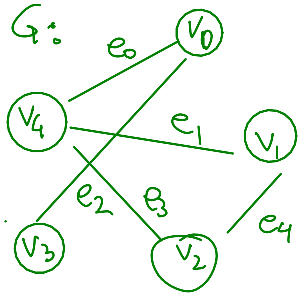
Lemma: Let Reduce(G,k) = (X,T).

(0) Reduction is polynomial time.

(i) If G has a subset V' of (exactly/~~at least~~) k vertices then X has a subset X' of sum T.

(ii) If X has a subset X' of sum T then G has a subset V' with (exactly/~~at least~~) k vertices.

G:



k=2

one column for every edge.

allows: put 1 in ith index if e_i is an edge from x_i

	e_0	e_1	e_2	e_3	e_4	
x_0:	1	0	1	0	0	1
x_1:	0	1	0	0	1	1
x_2:	.					
x_3:			.			
x_4:						1
y_0:	1	0	0	0	0	0
y_1:	0	1	0	0	0	0
y_2:	0	0	1	0	0	
y_3:	0	0	0	1	0	.
y_4:	0	0	0	0	1	0
T :	1	1	1	1	1	K

for each vertex ($x_0,\dots,x_4$); all one (last column)

for each edge ($y_0,\dots,y_4$); all zero (last column)

|V|+|E| element in S

Each element is represented using (|E|+1) numbers in base b.

The numbers are represented in base b. b is chosen so that sum of any column does not generate any carry in base b. For example b = |V|+1 = 6

∴ In this example, x_0 is $(101001)_6 = 7993$.

Let $T = (T)_{E+1} (T)_E (T)_{E-1} \dots (T)_3 (T)_2 (T)_1$ ((MSB) … (LSB))

Notation: $(x)_c$ denotes value in the c-th column of x when written in base c.

So $(x)_1$ = LSB of x.

Observe that $(y_c)_c = 1$.

Example

	MSB					LSB
base 6	1	0	1	0	0	1
" +	0	0	1	0	1	1
"	1	0	2	0	1	2
Column	6	5	4	3	2	1

Proof of ⇒ Suppose $v_1 \cdots v_k$ form an independent set in G.

Start with $S' = \{x_1 \cdots x_k\}$ [numbers corresponding to the vertices]

Claim: For each column c = 1(LSB), …, |E|+1 (MSB), $\sum_{x\in S'} (x)_c = (T)_c$

Proof for c = 1: The sum of LSBs of these (x)'s add up to K (LSB of T). since there are K x's in S' and LSB of each x is 1.

Proof for $c \geq 2$: Edge e_c has only two vertices. So only two x's have $(x)_c = 1$. Since x's in S' form an IS, either exactly one of them is in S' or none or none of them are in S'. If exactly one of them is in S', then $\sum_{x\in S'}(x)_c = 1 = (T)_c$.

If none are in S', include y_c in S'. Now $\sum_{x\in S'}(x)_c + (y)_c = 1$ ($\because (y_c)_c = 1$)

This y_c does not interfere with the claim for other c since $(y_c)_d = 0$ if $d \neq c$.

Proof of $\Leftarrow$: Suppose $S' = \{x_i, \cdots y_i, \cdots\}$ is a subset of S whose sum is T.

$\therefore$ For all column c $\sum_{z \in S'} (z)_c = (T)_c$.

Since $(T)_1 = k$ & $(y_i)_1 = 0$ & $(x_i)_1 = 1$, $\therefore$ S' must have exactly k x's.

Claim: These k x's correspond to k vertices which form an independent set.

Proof: Take any edge, let the corresponding column be $c \geq 2$.

Let the endpoints of that edge be x_i & x_j.

$\therefore$ $(x_i)_c = 1$ & $(x_j)_c = 1$. Since $(T_c) = 1$,

so, S' cannot have both x_i & x_j.

$\therefore$ For any edge, both its endpoints are not in S'.