Lecture 7

Jan 28, 2022

Problem set 3 is up on MS Teams

Last lecture: - Dynamic Programming as clever recursion
- Subset Sum

Today: 1. Subset Sum continued.
2. Knapsack problem

DP ≡ Recursive Substructure + Not too many subproblems

Recall: Subset Sum

Input: positive integers $a_1, a_2, \ldots, a_n, B$

Output: Does there exist a subset $S \subseteq \{1, \ldots, n\}$ s.t $\sum_{i \in S} a_i = B$

- Decision version
- Search version: Also output such a subset if it exists.

Intuition: suppose S is a solution. Then one of the following 2 cases must happen

① $a_n \in S \iff S \setminus \{a_n\}$ is a soln to the smaller instance $(a_1, \ldots, a_{n-1}; B - \{a_n\})$

② $a_n \notin S$ ⇒ S is a soln to the smaller instance $(a_1, \ldots, a_{n-1}; B)$

Natural Recursive Algorithm

1. Base case : instance of size 1 or 2
2. U = subsum $(a_1 \ldots a_{n-1}; B)$ ✓
3. If $a_n \leq B$,

   V = subsum $(a_1 \ldots a_{n-1}, B - a_n)$

   Else if $a_n == B$ then V = 1 } some case

   Else, V = 0
4. If U or V is 1 output 1, else output 0.

What is the running time? $T(n) \leq 2 \cdot T(n-1) + O(1)$

— Exponential in n.

Clearly a recursive substructure.
What about overlaps in subproblems?

Structure of smaller instances

$$(a_1, \ldots a_j ; \tilde{B})$$

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

$$\tilde{B} \in \{1, 2, \ldots, B\}$$

$n \cdot B$ instances in total.

Dynamic Programming: Intuition & pseudocode

1) Arranging the subproblems nicely and defining a recursive function.

(m, b) $\quad m \in \{1, 2, \dots, n\} - [n]$, $b \in \{1, 2, \dots, B\} - [B]$

↳ subset sum $\{a_1, \dots, a_m ; b\}$

$F(m,b) : [n] \times [B] \longrightarrow \{0,1\}$

$F(m,b) \equiv 1$ iff $\exists S \subseteq \{1, 2, \dots, m\}$ s.t $\sum_{i \in S} a_i = b$.

Goal: Decide $F(n, B)$

2. Handling the simple base cases

$$\begin{cases} F(m,0) = 0 & \forall m \in [n] \\ F(0,t) = 0 & \forall t \in [B] \\ F(m,t) = 0 & \forall t \leq 0. \\ \begin{cases} F(1,a_1) = 1 \\ F(1,b) = 0 & \forall b \in [B],\ b > a_1 \end{cases} \end{cases}$$

3. Establishing the recursion

$$\begin{cases} \forall m \geq 1,\ b \in \mathbb{N} \\ \quad F(m,b) = \max\left(F(m-1,b),\ F(m-1,b-a_m) \right) \end{cases}$$

4) Proof of the recursion | LHS ≤ RHS

$\Rightarrow F(m,b) = 1 \Rightarrow \max(F(m-1,b), F(m-1, b-a_m)) = 1$

LHS $F(m,b) = 1 \Rightarrow \exists S \subseteq [n]$ s.t $\sum_{i \in S} a_i = b$

Two cases arise

① $a_m \in S \Rightarrow F(m-1, b-a_m) = 1$

since $S - \{m\}$ satisfies $\sum_{j \in S-\{m\}} a_j = b - a_m$

② $a_m \notin S \Rightarrow F(m-1, b) = 1$

$\sum_{j \in S} a_j = 1$ and $S \subseteq \{1, \ldots, m-1\}$.

In both ① and ②, $\max(F(m-1,b), F(m-1, b-a_m)) = 1$.

Other direction

RHS $\subseteq$ LHS.

RHS = 0 — nothing to prove

RHS = 1. Want to show LHS = 1.

RHS = 1 $\Rightarrow$

① $F(m-1, b) = 1 \Rightarrow \exists S \subseteq [m-1],\ \sum_{j \in S} a_j = b$

take that $S \subseteq [m]$, so, $F(m,b) = 1$

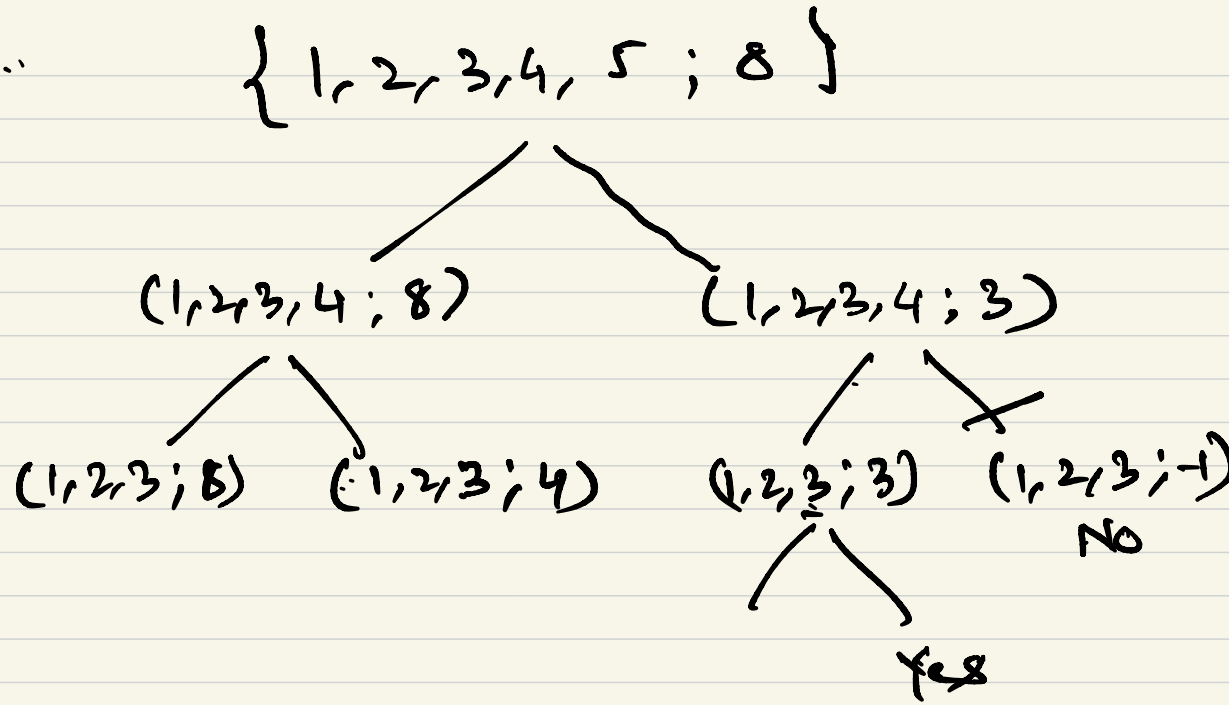
② $F(m-1, b-a_m) = 1$

$\Rightarrow \exists S \subseteq [m-1]$ s.t $\sum_{j \in S} a_j = b - a_m$ $\Big\}$ $(b - a_m \geq 0)$

Consider $\tilde{S} = S \cup \{m\}$. Note $\sum_{j \in \tilde{S}} a_j = b$.

$F(m,b) = 1$

Example:



5) Writing the pseudo code

1. Handle the Base cases of $F(n, b)$

2. For $1 \leq n \leq n$

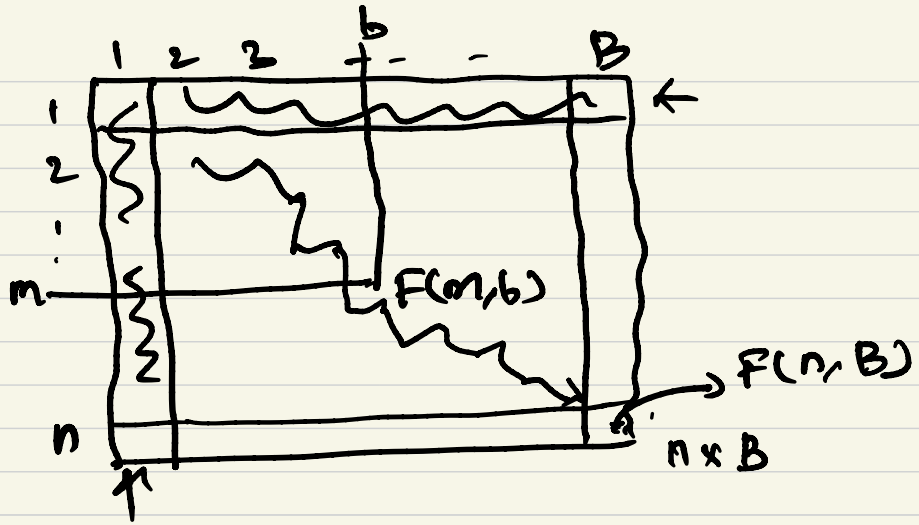
   For $1 \leq b \leq B$

   If $b - a_n < 0$, $F(n, b) = F(n-1, b)$

   Else,

   $$F(n, b) = \max(F(n-1, b), F(n-1, b-a_n))$$

Picture



Exercise: Design an algo for search version of Subset Sum.

Running time: $O(nB)$ → Is this any good?

↳ Is this efficient?

polynomial time in size of input.

Input size ?

- Unary: $\underbrace{11111}_{5}$ $\quad\underbrace{1111\cdots1}_{n}$
- binary: 101 $\quad$ binary encoding of n
- decimal

len(n) in unary $= n$

$$\left.\begin{array}{l}\text{len}(n) \text{ in binary} = \log_2 n \\ \text{len}(n) \text{ in decimal} = \log_{10} n\end{array}\right\} \Theta(\log n)$$

$$\text{len}(a_1 \cdots a_n ; B) = \left(\sum_{i=1}^{n} \log_2 a_i + \log_2 B\right)$$

$a_i \le a$.

$$\le (n \cdot \log a + \log_2 B)$$

Running time : $O(nB)$

- Is the running time polynomially bounded in input length?

No.

2: Knapsack Problem

Input: – n items

each item is specified by a pair (p_i, w_i) of positive integers

$p_i \rightarrow$ price of item i

$w_i \rightarrow$ weight of item i

– $B \in \mathbb{N}$

Output:

Find a subset $S \subseteq [n]$ that maximizes $\sum_{i \in S} p_i$ subject to $\sum_{j \in S} w_j \leq B$.

Naive algorithm:

- Try all subsets $S \subseteq [n]$
- Find the optimum

$\Big\} \; 2^n$

Recursive Structure : Intuition

Condition on item n (p_n, w_n)

suppose S is an optimal solution for $\{(p_i, w_i)_{i=1}^n, B\}$

① $n \in S \Rightarrow S \setminus \{n\}$ is an optimal solution for

$$\{(p_i, w_i)_{i=1}^{n-1}, B - w_n\}$$

② $n \notin S \Rightarrow S$ is an optimal solution for

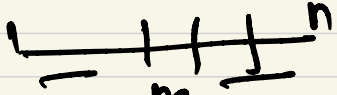
$$\{(p_i, w_i)_{i=1}^{n-1}, B\}$$

✓ So, some recursive structure to the problem.

✓ How many subproblems?

Subproblems are of the form

$$\{(p_i, w_i)_{i=1}^{m}, b\}$$



where $m \in [n]$

$b \in [B]$.

Recipe:

1. Define recursive fn, organize subproblems.

$F: [n] \times [B] \rightarrow$ power set of $\{1, \dots, m\}$.

$\rightarrow$ Power set of $[n]$

$F(m, b)$ = subset $S \subseteq [m]$ that maximizes $\sum_{i \in S} p_i$ subject to $\sum_{j \in S} w_j \leq b$.

2) Base cases —

$F(0, b) = \phi \quad 0 \leq b \leq B$

$F(m, 0) = \phi \quad 0 \leq m \leq n$

$F(m, b) = \phi \quad$ if $b < 0$.

3) Recursion

$$F(m, b) =$$

4) Prove the recursion

5) Pseudo code