Train VI

Oct 17. Ye Cao

DP ≈ "Careful" brute force
DP ≈ Right order (DAG) / Shortest path in some DAG

DP ≈ Guessing + recursion + memorization
Guessing: Take out some feature of the solution and exhaustively find all possibilities

DP Time = # subproblems * time / subproblem

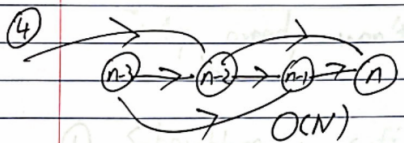
Treat Recursive call as O(1) because of using a memo
As opposed to mere brute force.

Last Time:
nth Fibonacci Number } DP in disguise
Shortest Path
How Bellman-ford come into place?
- DP.

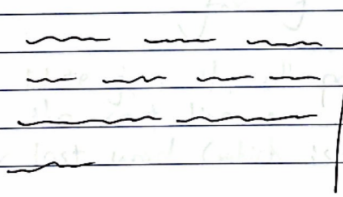
Today: Dynamic Programming II.
- 5 easy steps
- text justification
- perfect-information Blackjack
- patent pointers.

5 easy steps in DP: (Total Time)
① define subproblems → # subproblems
② guess (part of a solution) → # choices of guess
③ relate subproblem solutions → # time / subproblem (Simi-lar)
④ recursive & memorize OR build DP table bottom up
⑤ Solve original problem → Extra Time

Check subproblem recurrence is acyclic i.e. topological order.

	Fibonacci	Shortest Path
①	F_k for $k = 1 \dots n$ # subproblem = n	$\delta_k(s,v)$ for $v \in V$ $0 \le k < \lvert V \rvert$ # subproblems = V^2
②	1 (no guessing)	All edges in V (if any) Indegree (V)
③	$F_k = F_{k-1} + F_{k-2}$ $O(1)$	Similar $\delta_k(s,v) = \min\{\text{ind}\,\delta_{k-1}(s,x) + w(x,v) \mid (x,v) \in E\}$ $O(\text{indegree}(v))$
④	 $O(N)$	for $k = 0, 1, \dots, \lvert V \rvert - 1$ for $v \in V$ $O(VE)$
⑤	F_n $O(1)$	$\delta_{\lvert V \rvert - 1}(s,v)$ for $\forall v \in V$ $O(V^3)$

Text Justification:
Split text into "good" lines.



→ Locally Optimal
Microsoft word – (Non-latest) greedy approach
Latex – Dynamic Programming
Globally Optimal (not necessarily locally optimal)

text = list of words
badness (i, j)
use words $[i : j]$ as line

$$\text{badness}(i,j) = \begin{cases} (\text{page width} - \text{text width})^3 \\ \infty \quad \text{if don't fit} \end{cases}$$

Implication:

1. Highly discourage non-fit
2. If total width is less than page width, discourage.
3. Sometimes, you might want to slip little space for this line to do better on next line / globally (Global optimal rather than Local)
4. Problem is where to cut off the line?

Why greedy won't work always.

① Subproblem: suffixes word[i:]
- # subproblems n (# of words)

② Guess: Where to start 2nd line
- # choices ≤ n-1 O(n)

③ Recurrence:

$$DP[i] = \min(\text{badness}(i, j) + DP[j]) \text{ for } j \text{ in range}(i+1, n+1)$$

Note j is the all possible positions of where the next line can start — next word (i+1) til after last word (which is no word) (n+1)

Base case: $DP[n] = \emptyset$
Since last word is n-1 (0 indexed), the nth word won't generate any cost

Time / subproblem: O(n).

④ Topological order : $i = n, n-1, \cdots, \emptyset$

Compute backwards $O(n^2)$ → total time
Solve problem backwards

⑤ Original problem $DP[\emptyset]$ $O(1)$

Conclusion solved in quadratic time.

Parent Pointers :
remember which guess was best
Find the actual solution rather than the cost of best solution.

parent[i] = argmin_j (for j in range(i+1, n+1)
badness(i, j) + DP[j])

Follow parent pointers to find best solution

$\emptyset$ → parent[$\emptyset$] → parent[parent[$\emptyset$]]

where first line starts — where 2nd line starts — where 3rd line starts

As Long as topological order is known, we can always turn the resursive calls into array calls, hard part is figure out what to guess & subprob lem

Perfect - information Blackjack

- Deck: $C_0, C_1, C_2 \ldots, C_{n-1}$
- 1 player vs. dealer
- # 1 bet / hand

① subproblems: suffix of cards
at some point, start i-th card, want to start with i-th card on

② Guessing:
How many hits?
choices $\leq n$

③ recurrence:

$BJ[i] = \max($ outcome $\in \{-1, \phi, 1\} +$
$BJ[j]$
for # hits in range(n)
if valid play
↓
Check if point ≤ 21

$j = i + 4 +$ # hits + # dealer hits
↓ ↓
Guessing Part deterministic