COMP4128 Week 04 Tutorial

Yifan He

z5173587@unsw.edu.au
https://github.com/hharryyf/COMP4128-24T3-tutoring

Outline

We are doing Dynamic Programming this week

- Boredom (linear dp)
- Clear the String (interval dp)
- Pebbles (bitmask dp)

~~I'm very bad at dp. I always ask my teammates to solve dp problems in ICPC~~

Boredom

Given a sequence a consisting of n integers. The player can make several steps. In a single step he can choose an element of the sequence (let's denote it $a[k]$) and delete it, at that all elements equal to $a[k]-1$ and $a[k]+1$ also must be deleted from the sequence. That step brings $a[k]$ points to the player. What is the maximum number of points the player can get. ($n \leq 1e5$, $a[i] \leq 1e5$).

Example

N = 3, a = [1, 2, 3]
Answer: 4

Boredom

Analysis

- For each $a[k]$, it we remove it, we can get $a[k] \cdot cnt[a[k]]$ points

Boredom

Analysis

- For each $a[k]$, it we remove it, we can get $a[k] \cdot cnt[a[k]]$ points
- What is the subproblem?

Boredom

Analysis

- For each $a[k]$, it we remove it, we can get $a[k] \cdot cnt[a[k]]$ points
- What is the subproblem?
- Let dp[i] be the maximum points we can get when we are allowed to remove numbers in range 1 to i.

Boredom

Analysis

- For each $a[k]$, it we remove it, we can get $a[k] \cdot cnt[a[k]]$ points
- What is the subproblem?
- Let dp[i] be the maximum points we can get when we are allowed to remove numbers in range 1 to i.
- Base case: $dp[0] = 0$.

Boredom

Analysis

- For each $a[k]$, it we remove it, we can get $a[k] \cdot cnt[a[k]]$ points
- What is the subproblem?
- Let dp[i] be the maximum points we can get when we are allowed to remove numbers in range 1 to i.
- Base case: $dp[0] = 0$.
- The decision is to remove i or not remove i.

Boredom

Analysis

- For each $a[k]$, it we remove it, we can get $a[k] \cdot cnt[a[k]]$ points
- What is the subproblem?
- Let dp[i] be the maximum points we can get when we are allowed to remove numbers in range 1 to i.
- Base case: $dp[0] = 0$.
- The decision is to remove i or not remove i.
- Recursive case: $dp[i] = max(dp[i-1], dp[i-2] + cnt[i] \cdot i)$

Boredom

Analysis

- For each $a[k]$, it we remove it, we can get $a[k] \cdot cnt[a[k]]$ points
- What is the subproblem?
- Let dp[i] be the maximum points we can get when we are allowed to remove numbers in range 1 to i.
- Base case: $dp[0] = 0$.
- The decision is to remove i or not remove i.
- Recursive case: $dp[i] = max(dp[i-1], dp[i-2] + cnt[i] \cdot i)$
- Final answer: $dp[max(a[i], i = 1..N)]$

Clear the String

You are given a string s of length $N (N \le 500)$ consisting of lowercase Latin letters. You may apply some operations to this string: in one operation you can delete some continuous substring of this string, if all letters in the substring you delete are equal. Example: after deleting substring “bbbb” from string “abbbbaccdd” we get the string “aaccdd”. Calculate the minimum number of operations to delete the whole string s.

Example

N=5, s=“abaca”, ans=3
N=8, s=“abcddcba”, ans=4

Clear the String

Analysis

- Memory limit 256MB, time limit 3s
- What can we know from $N \leq 500$?

Clear the String

Analysis

- Memory limit 256MB, time limit 3s
- What can we know from $N \leq 500$?
- The dp state is 2-d
  - A 500*500*500 3-d int array would use more than 256MB
- The time complexity is $O(N^3)$
  - $N \leq 500$ is a good indication of an $O(N^3)$ solution

Clear the String

Analysis (cont.)

- What kind of dp has a 2-d state?

Clear the String

Analysis (cont.)

- What kind of dp has a 2-d state?
- Interval dp!

Clear the String

Analysis (cont.)

- What kind of dp has a 2-d state?
- Interval dp!
- Subproblem: $dp[l][r]$ to be the minimum number of operations to remove the substring $s[l:r]$
- Final answer: $dp[1][N]$
- Base case: $dp[i][j] = 1$ if $i = j$; $dp[i][j] = 0$ if $i > j$.
- The hard part is the recursive case

Clear the String

Recursive case

- Once we remove part of a string, the remaining string merges together

Clear the String

Recursive case

- Once we remove part of a string, the remaining string merges together
- We want to solve $dp[l][r]$
- Consider the left most character $s[l]$
  1. We either remove it directly with 1 operation, or
  2. We remove it together with some $s[k] = s[l], l < k \leq r$
- For case 1:

Clear the String

Recursive case

- Once we remove part of a string, the remaining string merges together
- We want to solve $dp[l][r]$
- Consider the left most character $s[l]$
  1. We either remove it directly with 1 operation, or
  2. We remove it together with some $s[k] = s[l], l < k \leq r$
- For case 1:
  - $dp[l][r] = 1 + dp[l+1][r]$
- For case 2:

Clear the String

Recursive case

- Once we remove part of a string, the remaining string merges together
- We want to solve $dp[l][r]$
- Consider the left most character $s[l]$
  1. We either remove it directly with 1 operation, or
  2. We remove it together with some $s[k] = s[l], l < k \leq r$
- For case 1:
  - $dp[l][r] = 1 + dp[l+1][r]$
- For case 2:
  - $dp[l][r] = dp[l+1][k] + dp[k+1][r]$
- $dp[l][r] = min\{1 + dp[l+1][r], min(dp[l+1][k] + dp[k+1][r], s[l] = s[k])\}$

Demo

Pebbles

You are given an unlimited number of pebbles to distribute across an N ×N game board ($1 \leq N \leq 15$), where each square on the board contains an integer point value between 1 and 99, inclusive. The integers on a given board may not be unique. The player distributes pebbles across the board so that: At most one pebble resides in any given square. No two pebbles are placed on adjacent squares. Two squares are considered adjacent if they are horizontal, vertical, or diagonal neighbors. The goal is to maximize the number of points claimed by your placement of pebbles.

Example

71	24	95	56	54
85	50	74	94	28
92	96	23	71	10
23	61	31	30	46
64	33	32	95	89

Pebbles

Analysis

- N is pretty small, 2^{15} is only 32768

Pebbles

Analysis

- N is pretty small, 2^{15} is only 32768
- Bitmask dp!

Pebbles

Analysis

- N is pretty small, 2^{15} is only 32768
- Bitmask dp!
- If a row has a mask representation m, we select the $a[i][j]$ if and only if the j-th bit of m is 1.

Pebbles

Analysis

- N is pretty small, 2^{15} is only 32768
- Bitmask dp!
- If a row has a mask representation *m*, we select the $a[i][j]$ if and only if the *j*-th bit of *m* is 1.
- Subproblem: $dp[i][m]$ is the maximum number of points if we only consider the first i rows and row number i has a mask *m*.

Pebbles

Analysis

- N is pretty small, 2^{15} is only 32768
- Bitmask dp!
- If a row has a mask representation m, we select the $a[i][j]$ if and only if the j-th bit of m is 1.
- Subproblem: $dp[i][m]$ is the maximum number of points if we only consider the first i rows and row number i has a mask m.
- Not all $m = 0..2^N - 1$ are valid. For example, the binary representation $m = 3$ is 11, invalid!

Pebbles

Analysis

- N is pretty small, 2^{15} is only 32768
- Bitmask dp!
- If a row has a mask representation m, we select the $a[i][j]$ if and only if the j-th bit of m is 1.
- Subproblem: $dp[i][m]$ is the maximum number of points if we only consider the first i rows and row number i has a mask m.
- Not all $m = 0..2^N - 1$ are valid. For example, the binary representation $m = 3$ is 11, invalid!
- We should only keep m such that its binary representation has no two consecutive 1s.

Pebbles

Analysis

- How to check a number m has no two consecutive 1s in its binary representation?

Pebbles

Analysis

- How to check a number m has no two consecutive 1s in its binary representation?
- $m \,\&\, (m << 1) = 0$

Pebbles

Analysis

- How to check a number m has no two consecutive 1s in its binary representation?
- $m \,\&\, (m << 1) = 0$
- Subproblem: $dp[i][m]$ is the maximum number of points if we only consider the first i rows and row number i has a mask m. If m is invalid, $dp[i][m] = -\infty$

Pebbles

Analysis

- How to check a number m has no two consecutive 1s in its binary representation?
- $m \,\&\, (m << 1) = 0$
- Subproblem: $dp[i][m]$ is the maximum number of points if we only consider the first i rows and row number i has a mask m. If m is invalid, $dp[i][m] = -\infty$
- Base case: $dp[0][m] \; = \; \sum_{(m>>j)\&1=1} a[0][j]$

Pebbles

Analysis

- How to check a number m has no two consecutive 1s in its binary representation?
- $m \,\&\, (m << 1) = 0$
- Subproblem: $dp[i][m]$ is the maximum number of points if we only consider the first i rows and row number i has a mask m. If m is invalid, $dp[i][m] = -\infty$
- Base case: $dp[0][m] = \sum_{(m>>j)\&1=1} a[0][j]$
- Recursive case: $dp[i][m] = max(dp[i-1][m'] + \sum_{(m>>j)\&1=1} a[i][j], \; m' \; is \; valid)$

Pebbles

Analysis

- Even if m' is valid, and m is valid, if m' is the bit representation of row $i-1$, and m is the bit representation of row i, picking m' and m together might not be valid!

Pebbles

Analysis

- Even if m' is valid, and m is valid, if m' is the bit representation of row $i-1$, and m is the bit representation of row i, picking m' and m together might not be valid!
- m’ = 101, m = 010, shared diagonal!

Pebbles

Analysis

- Even if m' is valid, and m is valid, if m' is the bit representation of row $i-1$, and m is the bit representation of row i, picking m' and m together might not be valid!
- m’ = 101, m = 010, shared diagonal!
- Cannot put pebbles on $a[i-1][j]$ and $a[i][j]$ for some i, j
- m’ and m cannot both have 1 on the j-th bit for some $0 \leq j < N$

Pebbles

Analysis

- Even if m' is valid, and m is valid, if m' is the bit representation of row $i-1$, and m is the bit representation of row i, picking m' and m together might not be valid!
- m' = 101, m = 010, shared diagonal!
- Cannot put pebbles on $a[i-1][j]$ and $a[i][j]$ for some i, j
- m' and m cannot both have 1 on the j-th bit for some $0 \leq j < N$
- $m \,\&\, m' = 0$

Pebbles

Analysis

- Cannot put pebbles on $a[i-1][j]$ and $a[i][j+1]$ for some i, j

Pebbles

Analysis

- Cannot put pebbles on $a[i-1][j]$ and $a[i][j+1]$ for some i, j
- Cannot put pebbles on $a[i-1][j]$ and $a[i][j-1]$ for some i, j

Pebbles

Analysis

- Cannot put pebbles on $a[i-1][j]$ and $a[i][j+1]$ for some i, j
- Cannot put pebbles on $a[i-1][j]$ and $a[i][j-1]$ for some i, j
- $m \,\&\, (m' << 1) = 0$
- $(m << 1) \,\&\, m' = 0$

Pebbles

Conclusion

- $dp[i][m] = max(dp[i-1][m'] + \sum_{(m>>j)\&1=1} a[i][j],\ m' \textit{ is valid})$
- $m \,\&\, m' = 0$
- $m \,\&\, (m' << 1) = 0$
- $(m << 1) \,\&\, m' = 0$

Time complexity

- $O(N \cdot |A|^2)$, $A = \{k | k \in [0, 2^N - 1], k \textit{ is valid}\}$
- $N = 15$, $|A| = 1597$

Demo