

956. Tallest Billboard II DP II Reducing DP States II Hashing II 0/1

Knapsack

24 June 2023 07:19 AM

956. Tallest Billboard

Hard 903 29

Companies

You are installing a billboard and want it to have the largest height. The billboard will have two steel supports, one on each side. Each steel support must be an equal height.

You are given a collection of rods that can be welded together. For example, if you have rods of lengths 1, 2, and 3, you can weld them together to make a support of length 6.

Return the largest possible height of your billboard installation. If you cannot support the billboard, return 0.

Example 1:

Input: rods = [1,2,3,6]
Output: 6
Explanation: We have two disjoint subsets {1,2,3} and {6}, which have the same sum = 6.

Example 2:

Input: rods = [1,2,3,4,5,6]
Output: 10
Explanation: We have two disjoint subsets {2,3,5} and {4,6}, which have the same sum = 10.

What we can see from above diagram → For Every Rod we have 3 Options

1. (Not take it) ← Standard 0/1 Knapsack problem

2. (Take it) → Put in Support 1

3. (Take it) → Put in Support 2

DSA Topics Wise Important Problems

Top 150 Interview Questions by Aryan

Complete DP Problem Solving by Aryan

OF WAYS TO REORDER ARRAY TO BE STRICTLY INCR

Complete Mathematics & Number Theory Problems by...

Complete Array Problem Playlist by Aryan

Complete Graph Problem Playlist by Aryan

Complete Linked List by Aryan

Playlists: Aryan Mittal Updated 2 days ago View full playlist

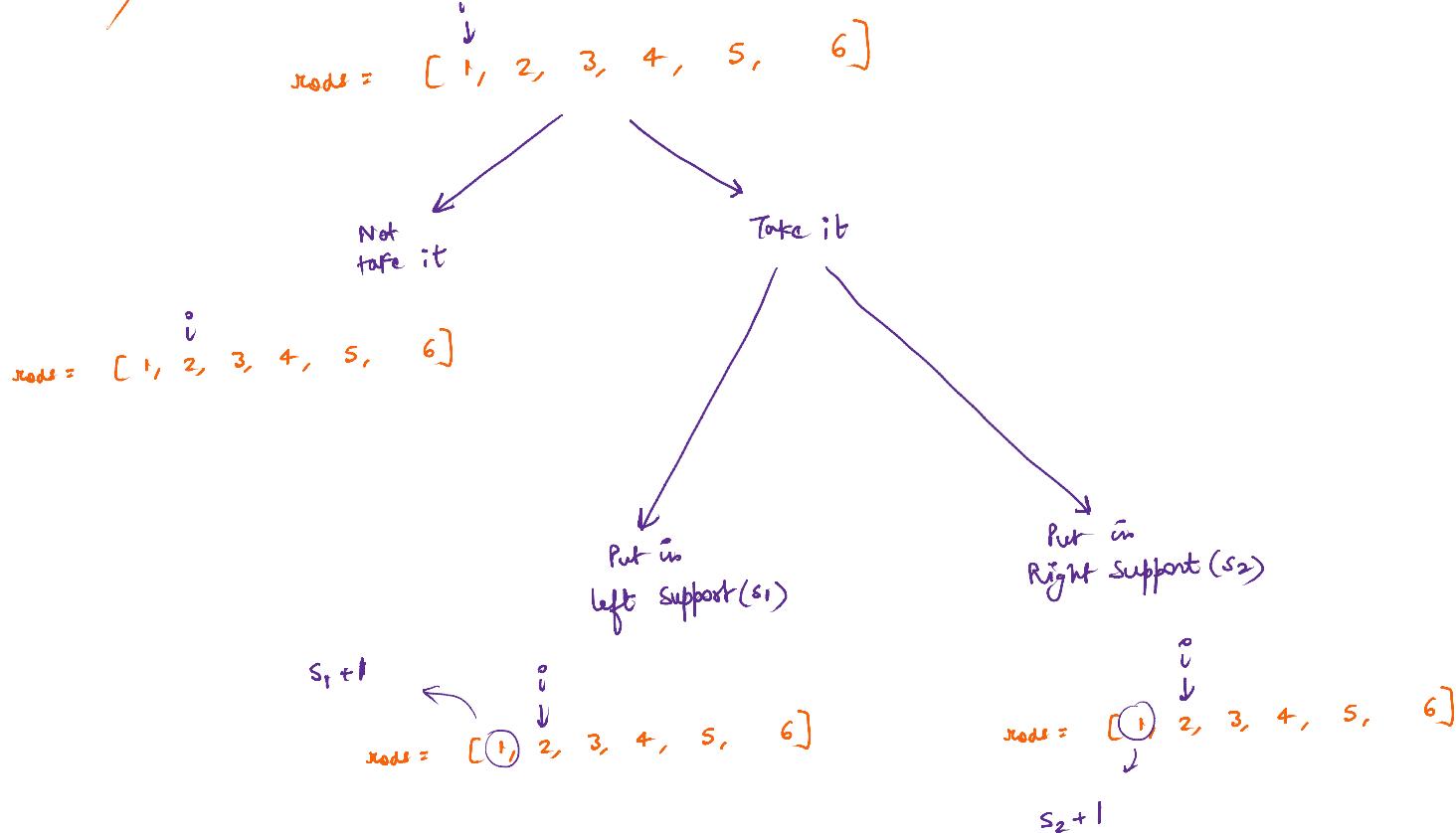
Playlists: Aryan Mittal Updated 6 days ago View full playlist

Playlists: Aryan Mittal Updated 7 days ago View full playlist

Playlists: Aryan Mittal Updated 6 days ago View full playlist

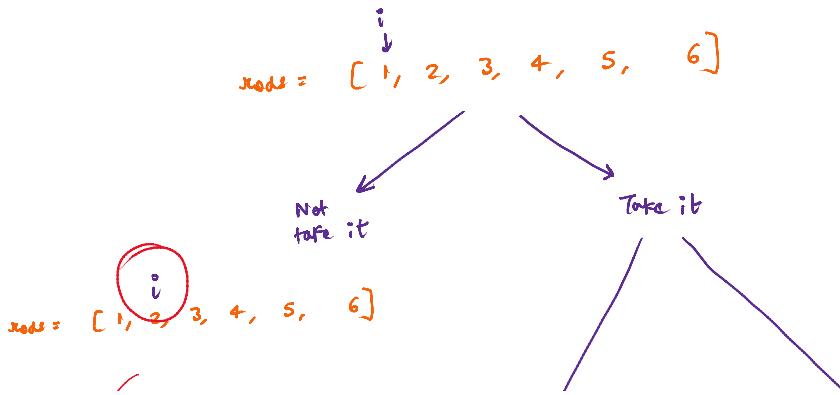
Playlists: Aryan Mittal Updated 5 days ago View full playlist

⇒ For every Rod → we have 3 Options ..

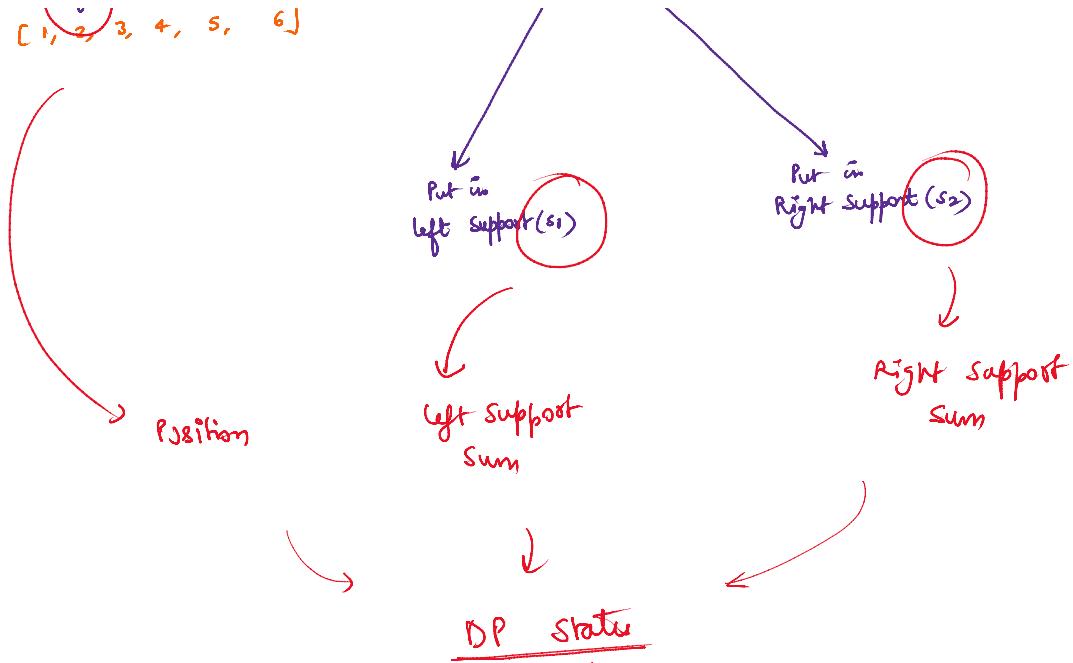


↳ we can see from the Tree itself it is a Recursion ↳ kind of same
q1 knapsack
Recursion trees
will be made

Memoization
by DP ← Repeating
Sub-problems



$\text{rods} = [1, 2, 3, +, 5, 6]$



Simple Recursion

```

class Solution{
    int ans, n;
    vector<int> rods;
public:
    int tallestBillboard(vector<int>& rods) {
        this->rods = rods;
        n = rods.size();
        ans = 0;

        return solve(0, 0, 0);
    }
    int solve(int i, int s1, int s2) {
        if (i == n) {
            if (s1 == s2) return s1; // If both support are equal
            return 0;
        }

        int ans = 0;
        int opt1 = solve(i + 1, s1, s2); // Don't take that rod, add in support
        int opt2 = solve(i + 1, s1 + rods[i], s2); // Take Rod, add in support 1
        int opt3 = solve(i + 1, s1, s2 + rods[i]); // Take Rod, add in support 2

        return ans = max({opt1, opt2, opt3});
    }
};

```

Annotations on the code:

- Position**: Points to the parameter *i* in the `solve` function.
- left support sum**: Points to the parameter *s1* in the `solve` function.
- right support sum**: Points to the parameter *s2* in the `solve` function.
- As we read end**: Points to the condition `i == n`.
- If both support are equal**: Points to the condition `s1 == s2` in the `return s1;` statement.
- Don't take that rod & move on**: Points to the `opt1` calculation.
- Take Rod, add in support 1**: Points to the `opt2` calculation.
- Take Rod, add in support 2**: Points to the `opt3` calculation.
- max equal support**: Points to the `max` operation at the bottom of the `solve` function.

```

class Solution{
    int ans, n;
    vector<int> rods;
public:
    int tallestBillboard(vector<int>& rods) {
        this->rods = rods;
        n = rods.size();
        ans = 0;

        return solve(0, 0, 0);
    }

    int solve(int i, int s1, int s2) {
        if (i == n) {
            if (s1 == s2) return s1;
            return 0;
        }

        int ans = 0;

        int opt1 = solve(i + 1, s1, s2);
        int opt2 = solve(i + 1, s1 + rods[i], s2);
        int opt3 = solve(i + 1, s1, s2 + rods[i]);

        return ans = max({opt1, opt2, opt3});
    }
};

```

if we memoize this by

$dp[i][s1][s2]$

Constraints:

- $1 \leq rods.length \leq 20$
- $1 \leq rods[i] \leq 1000$
- $\sum(rods[i]) \leq 5000$

$$= 20 + 5000 * 5000$$

$$= 50 * 10^7 \approx 10^8$$

both time & space
not feasible in
with DP too.

going to next position

optimize DP state :-

$dp[i][s1][s2]$

Identify similarity in DP state

sum of support 1

sum of support 2

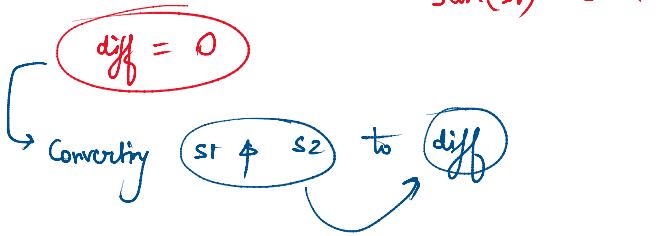
Ultimately want

$$\text{Sum}(s1) = \text{Sum}(s2)$$

or
it can take
difference
also

$$s1 - s2 = 0$$

$$\text{Sum}(s1) - \text{Sum}(s2) = 0$$



$$0 \leq s_1 \leq 5000$$

$$0 \leq s_2 \leq 5000$$

$$\begin{aligned} -5000 &\leq s_1 - s_2 \leq 5000 \\ (0 - 5000) &\quad (5000 - 0) \end{aligned}$$

$$-5000 \leq diff \leq 5000 \Rightarrow$$

$$\hookrightarrow offset = 5000$$

$$0 \leq diff \leq 2 * 5000$$

$$\downarrow$$

$$s_1 - s_2$$

```

const int MIN = -1e4, offset = 5000;
class Solution {
    int ans, n;
    vector<int> rods;
    int dp[21][2 * offset + 1];
public:
    int tallestBillboard(vector<int>& rods) {
        this->rods = rods;
        n = rods.size();

        memset(dp, -1, sizeof(dp));
        int ans = solve(0, 0);
        if (ans < 0) return 0;
        else return ans;
    }

    int solve(int i, int diff) {
        if (i == n) {
            if (diff == 0) return 0;
            return INT_MIN;
        }

        int &ans = dp[i][diff + offset];
        if (ans != -1) return ans;

        int opt1 = solve(i + 1, diff);
        int opt2 = rods[i] + solve(i + 1, diff + rods[i]);
        int opt3 = solve(i + 1, diff - rods[i]);

        return ans = max({opt1, opt2, opt3});
    }
};

DP Memoization step

```

initially both are 0

i *diff* $= s_1 - s_2$

No possible solution

reached end

already adding values..

Don't take it

take it in s_1

$$= a + rods(i) - b$$

$$= a - b + rods(i) = diff + rods(i)$$

take it in s_2

$$= a - (b + rods(i))$$

$$= a - b - rods(i)$$

$$= diff - rods(i)$$

Addig only in
1 option to not
Count that Twice

CODE

```

const int MIN = -1e4, offset = 5000;
class Solution {
    int ans, n;
    vector<int> rods;
    int dp[21][2 * offset + 1];
public:
    int tallestBillboard(vector<int>& rods) {

```

```

this->rods = rods;
n = rods.size();
memset(dp , -1 , sizeof(dp));
int ans = solve(0, 0);
if (ans < 0) return 0;
else return ans;
}
int solve(int i, int diff) {
    if (i == n) {
        if (diff == 0) return 0;
        return INT_MIN;
    }
    int &ans = dp[i][diff + offset];
    if (ans != -1) return ans;
    int opt1 = solve(i + 1 , diff);
    int opt2 = rods[i] + solve(i + 1 , diff +
rods[i]);
    int opt3 = solve(i + 1 , diff - rods[i]);
    return ans = max({opt1, opt2, opt3});
}
};

```

