

0-1 Knapsack

You are packing for a vacation on the sea side and you are going to carry only one bag with capacity S ($1 \leq S \leq 1000$).

You also have N ($1 \leq N \leq 1000$) items that you might want to take with you to the sea side. Unfortunately you can not fit all of them in the knapsack so you will have to choose.

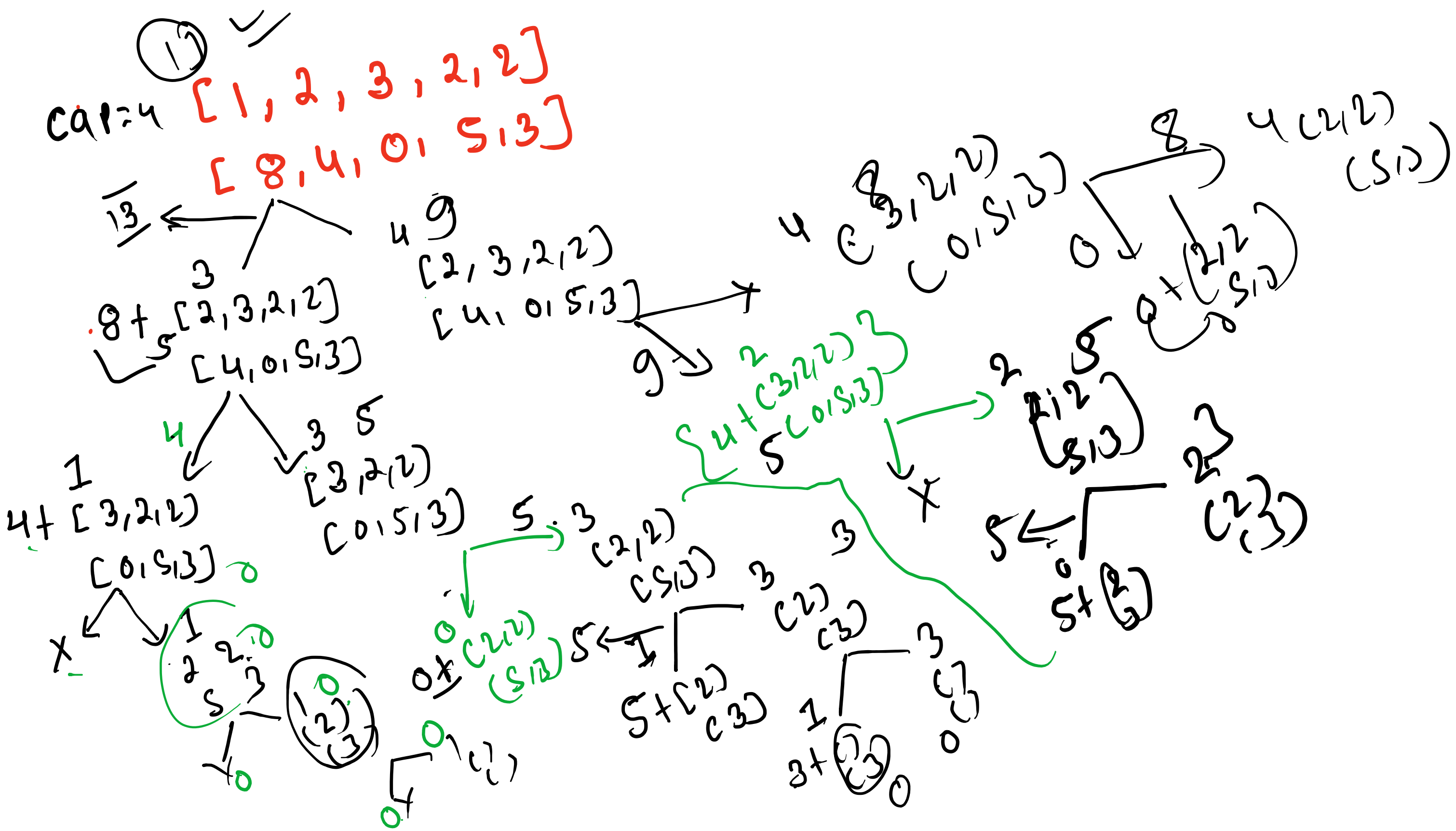
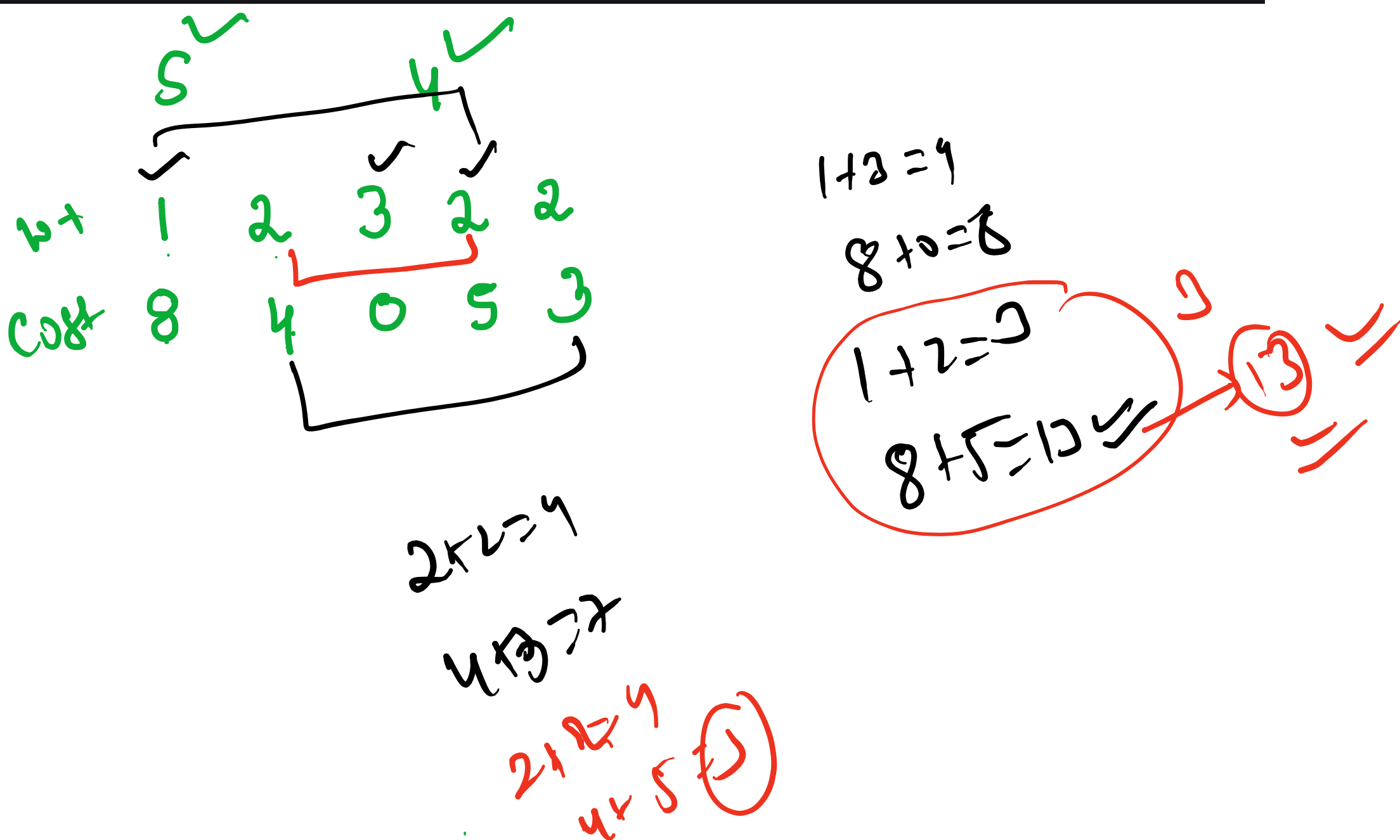
For each item you are given its size and its value.

You want to maximize the total value of all the items you are going to bring.

What is this maximum total value?

... ..

5 4
1 2 3 2 2
8 4 0 5 3



```

public static int Knapsack(int[] wt, int[] val, int cap, int idx) {
    if (cap == 0 || idx == wt.length) {
        return 0;
    }
    int inc = 0, exc = 0;
    if (cap >= wt[idx]) {
        inc = val[idx] + Knapsack(wt, val, cap - wt[idx], idx + 1);
    }
    exc = Knapsack(wt, val, cap, idx + 1);
    return Math.max(inc, exc);
}

```

Handwritten notes on the code:

- At the top right: $W(n-1, 5) \leftarrow (8, 1)$ and $APR \rightarrow 40x$ with a box containing $200+10$, $200+10$, and $80+10$.
- Next to the `if (cap == 0 || idx == wt.length)` line: \uparrow with $12 \dots$ written next to it.
- Next to the `if (cap >= wt[idx])` line: A red arrow points to a circle containing inc , which then points to a circle containing val . Below this, another red arrow points to a circle containing DP .
- At the bottom: A red arrow points to the `return Math.max(inc, exc);` line, with a red circle containing 7 next to it.
- At the bottom right: Two arrays are written: $[2, 3, 2, 4]$ and $[5, 7, 0, 3]$.
- At the bottom center: A red number 4 is written.

