Hands-on:

- https://nus.kattis.com/problems/canvas
- You have about 30 minutes to try coding an AC solution
- Lab TA will give gradual hints per 5m interval and live code that hint
- Full AC solution **will not** be given,
  the last hint will be something that is “near AC”
- If you get AC before the last hint is given, Lab TA will recognize you and will count that as a factor to decide the “lab participation points”
- Albeit not graded, you are encouraged to continue working until you get AC after all these hints are poured…

Canvas Painting Summary (after 5m)

Constraints:

$1 \le N \le 100K$ (read: BIG)

$1 \le S_i \le 100K$

Multiple TC!

$1 \le T \le 100$

Sum(N) of all TC in a file $\le 100K$

- Given an array S which contains the size of canvases of the same color.
- Initially, you can rearrange them in any order that you want.
- Afterward, you can do the following steps repeatedly:
    - Choose one color that exists in the array. Assume that there are M canvases of that color.
    - Choose X first canvas(es) from the left of that color ($1 \le X < M$).
    - Give them a color that has never existed in that set.
    - Give the rest of those M canvases another non-present color.
- The cost of this operation is the sum of size from those M canvases.
- Determine the minimum cost such that every canvas has a unique color.
- Pictorial Explanation is Better – see next slide

Canvas Painting Summary (after 5m)

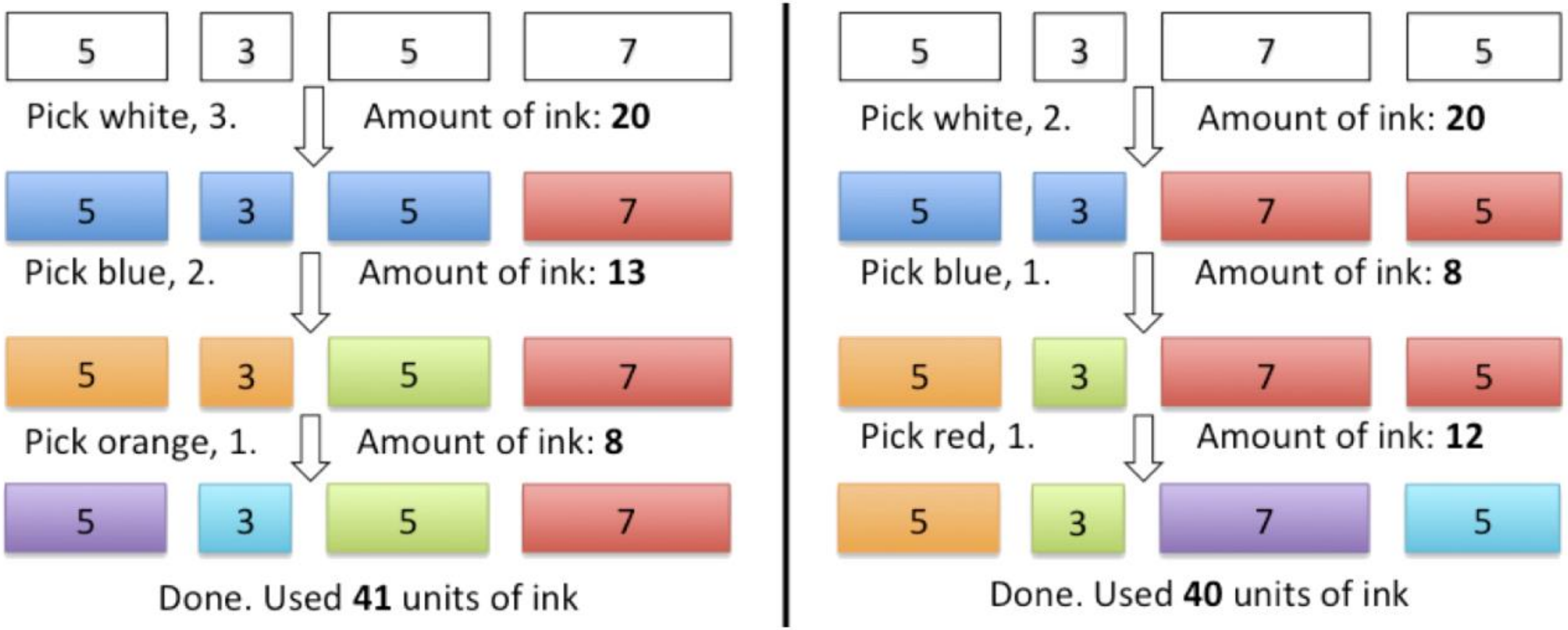

image source: kattis.com

Canvas Painting (after 10m) – major hint :O…

Let’s reverse the order of thinking…

- Assume that having the same color means that two canvases are in the same group.
- We know that in end, there are **N** groups,
  each contains one canvas.
- We want to merge those canvases into one group.
- We can merge two groups at a time,
  cost: sum of canvas sizes in those groups.

Canvas Painting (after 10m) – major hint :O…

Key Observation

Always merge two groups with the *smallest total size* at every step!

This is called "a greedy strategy"
(in CS3230, you have to "prove" the correctness,
here in CS2040C, just implement it)

A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage[1] with the intent of finding a global optimum

Greedy algorithms determine minimum number of coins to give while making change.
These are the steps a human would take to emulate a greedy algorithm to represent 36 cents using only coins with values {1, 5, 10, 20}.
The coin of the highest value, less than the remaining change owed, is the local optimum

https://en.wikipedia.org/wiki/Greedy_algorithm

Canvas Painting (after 15m) – the details

How does it work?

1. Maintain a set of groups that we have. Initially there N groups.
2. At every step, choose 2 groups with smallest sizes and remove them from the set.
3. Let W be their total size. Our answer increases by W.
4. Insert a new group with size equal to W.
5. Repeat from 2 until size of the set equals to 1.

Easy way to maintain the set?

priority_queue

Canvas Painting (after 20m) – the pseudo-code

Pseudo-code

```
#define LL   long long
priority_queue<LL, vector<LL>, greater<LL>> pq;

...

LL ans = 0LL;
while (!pq.empty()) {
     LL a = pq.top(); pq.pop();
     LL b = pq.top(); pq.pop();
     ans += a + b;
     pq.push(a + b);
}
```

Canvas Painting (summary slides at the end)

Time Complexity

O(log(N))	Push/pop from PQ
N	Number or push/pop
O(NlogN)	**Total**

Memory Space

N	Max size of PQ
O(N)	**Total**