MysticAndCandies

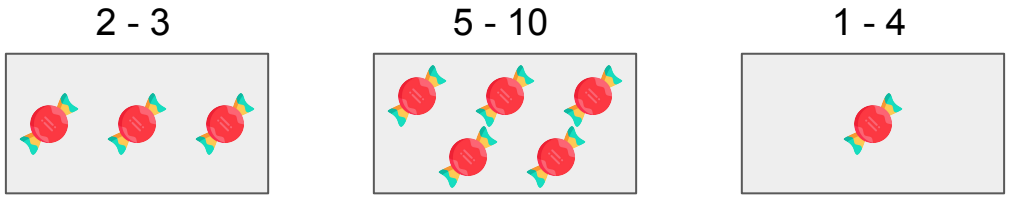
Samantha Zimmermann

COSC 494

Fall 2020

The Problem

- mystic_tc wants to eat a certain number of candies
- Many closed boxes with random # of candies in each
- Open minimum # of boxes
- Must eat all candies in every box opened
- The catch: only know ranges in each box



The Method

X: # candies to eat
C: total candies
N: number of boxes

- Class name: MysticAndCandies
- Method name: minBoxes
- Parameters:

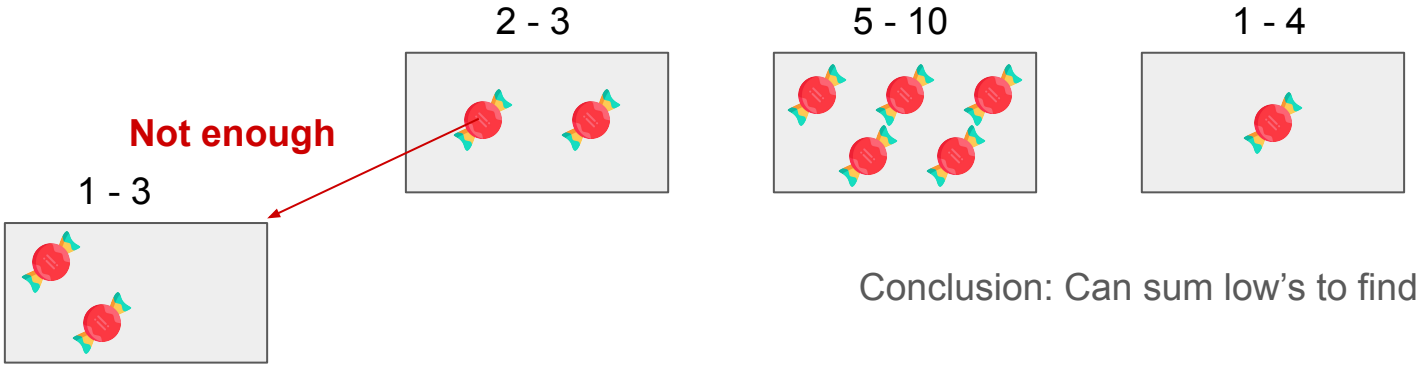
C	`int`	Total # of candies	Sum(Lows) ≤ C ≤ Sum(Highs), 1 ≤ C ≤ 50
X	`int`	# of candies to eat	1 ≤ X ≤ C
low	`int[]`	Lower bounds	low[i] ≤ high[i], size(high) = size(low)
high	`int[]`	Upper bounds	… 1 ≤ size(high) ≤ 10,000,000

- Returns: `int`
- Signature: `int MysticAndCandies::minBoxes(int C, int X, int[] low, int[] high)`

The Solution

X: # candies to eat
C: total candies
N: number of boxes

- Consider:
  - mystic_tc chooses a solution
  - # candies in solution ≥ X
  - To guarantee the above, one of two cases →
    1. Boxes in solution are at their min capacity



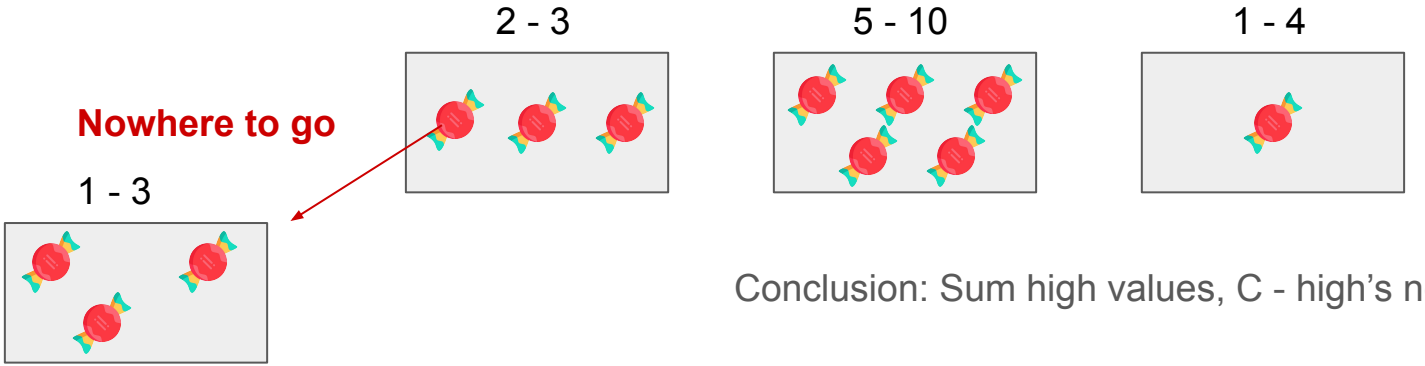
Conclusion: Can sum low's to find # boxes

The Solution

X: # candies to eat
C: total candies
N: number of boxes

- Consider:
  - mystic_tc chooses a solution
  - # candies in solution ≥ X
  - To guarantee the above, one of two cases →

2. Boxes outside solution are at their max capacity



Conclusion: Sum high values, C - high's not in the solution

The Solution

X: # candies to eat
C: total candies
N: number of boxes

```
Sort low's in descending order
Sort high's in ascending order
For i from 0 to N:
    sum += low[i]
    If sum ≥ X, minBoxes = i+1
For i from 0 to N:
    sum += high[i]
    If C-sum ≤ X, minBoxes = N-i-1
Return smaller minBoxes between the two
```

Subset of boxes at minimum capacity

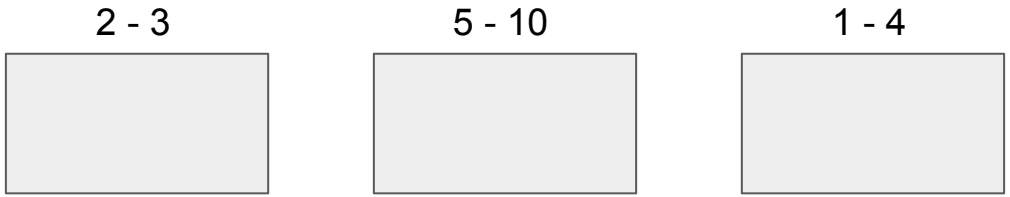
Rest of boxes at maximum capacity

An Example

X: # candies to eat
C: total candies
N: number of boxes

- **X = 5, C = 15**
  1. Sort low: [2, 5, 1] → [5, 2, 1]
  2. Sort high: [3, 10, 4] → [3, 4, 10]
  3. Sum low's: 5, 5 + 2 = 7, 5 + 2 + 1 = 8 → [5, 7, 8]

5 ≥ X → **minBoxes = 1** (i = 0)

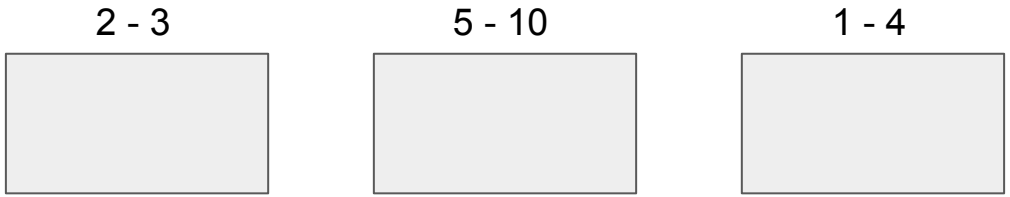


Low: [2, 5, 1]
High: [3, 10, 4]

An Example

X: # candies to eat
C: total candies
N: number of boxes

- **X = 9, C = 15**
  1. Sort low: → [5, 2, 1]
  2. Sort high: → [3, 4, 10]
  3. Sum low's: → 5, 5 + 2 = 7, 5 + 2 + 1 = 8 → [5, 7, 8] → **None are enough!**
  4. Sum high's: 15 - 3 = 12, 15 - (3 + 4) = 8, 15 - (3 + 4 + 10) = -2 → [12, 8, -2]
     8 ≤ X → **minBoxes = 1** (i = 1)



Low: [2, 5, 1]
High: [3, 10, 4]

The Complexity

X: # candies to eat
C: total candies
N: number of boxes

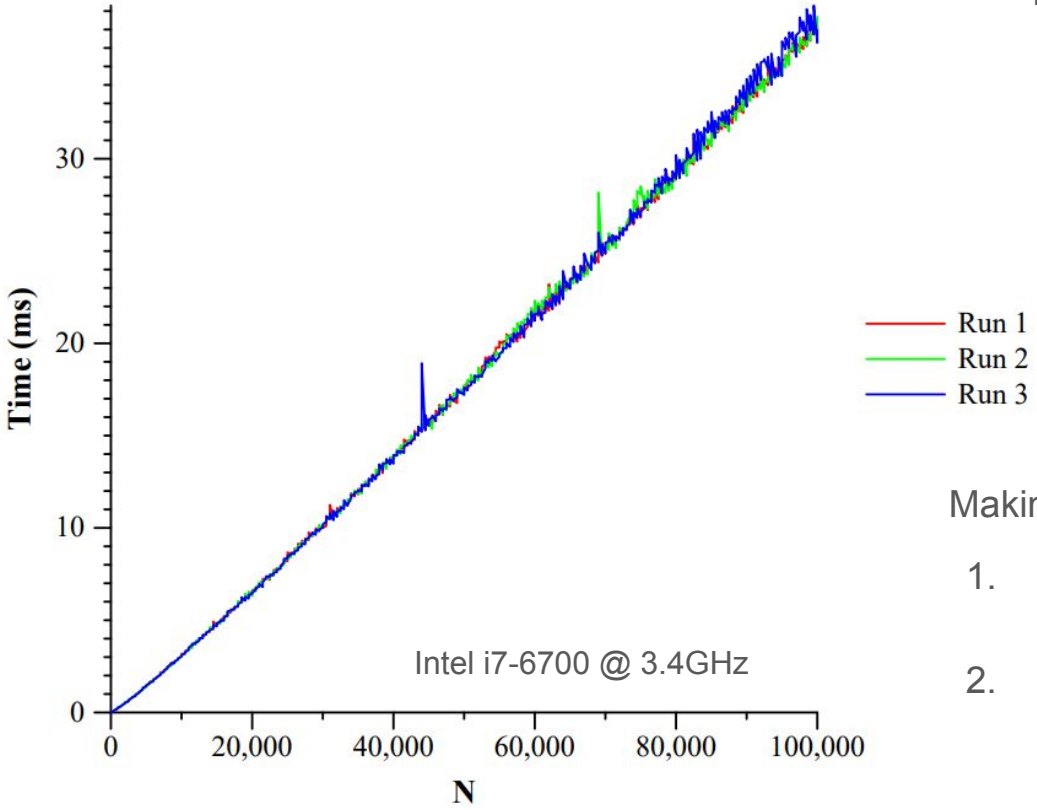
```
Sort low's in descending order
Sort high's in ascending order
For i from 0 to N:
    Sum += low[i]
    If sum ≥ X, minBoxes = i+1
For i from 0 to N:
    Sum += high[i]
    If C-sum ≤ X, minBoxes = N-i-1
Return smaller minBoxes between the two
```

std::sort → O(N log N) average case

for i from 0 to N → O(N)

→ O(2N + 2N log N) → **O(N log N)**

The Data



X: # candies to eat
C: total candies
N: number of boxes

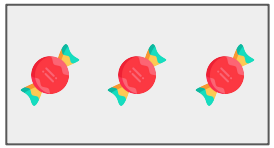
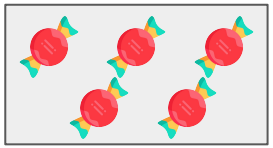
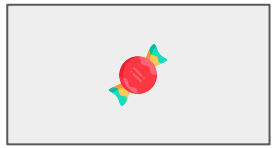
→ O(N log N)

Making it faster:

1. Use a priority queue instead of sorting
   → std::priority_queue O(N) from vector
2. Base case C = X
   → Gets rid of worse case

The Statistics

- 742 people opened
- 501 people submitted a solution (67.5%)
- 73.05% of submissions were correct
- Best score: 293.0
- Average score: 199.05
- Best time: 4:40 (5:39 in C++)
- Average time: 26:27

MysticAndCandies

Samantha Zimmermann

COSC 494

Fall 2020