Greedy Algorithms: Grouping Children

Michael Levin

Higher School of Economics

Algorithmic Toolbox
Data Structures and Algorithms

Outline

1. The Problem
2. Naive Algorithm
3. Efficient Algorithm

Many children came to a celebration. Organize them into the minimum possible number of groups such that the age of any two children in the same group differ by at most one year.

Outline

1. The Problem
2. Naive Algorithm
3. Efficient Algorithm

MinGroups(C)

```
m ← len(C)
for each partition into groups
C = G_1 ∪ G_2 ∪ ··· ∪ G_k:
  good ← true
  for i from 1 to k:
    if max(G_i) − min(G_i) > 1:
      good ← false
  if good:
    m ← min(m, k)
return m
```

Running time

Lemma

The number of operations in `MinGroups`(C) is at least 2^n, where n is the number of children in C.

Proof

- Consider just partitions in two groups
- $C = G_1 \cup G_2$
- For each $G_1 \subset C$, $G_2 = C \setminus G_1$
- Size of C is n
- Each item can be included or excluded from G_1
- There are 2^n different G_1
- Thus, at least 2^n operations $\square$

Asymptotics

- Naive algorithm works in time $\Omega(2^n)$
- For $n = 50$ it is at least

$$2^{50} = 1125899906842624$$

operations!
- We will improve this significantly

Outline

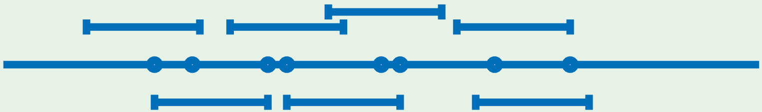
1. The Problem
2. Naive Algorithm
3. Efficient Algorithm

Covering points by segments

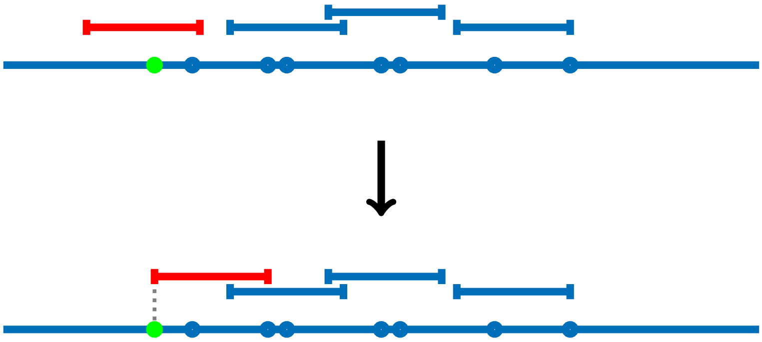
Input: A set of n points $x_1, \ldots, x_n \in \mathbb{R}$.

Output: The minimum number of segments of unit length needed to cover all the points.

Example



Safe move: cover the leftmost point with a unit segment which starts in this point.



Assume $x_1 \le x_2 \le \ldots \le x_n$

```
PointsCoverSorted(x_1, ..., x_n)

R ← {}, i ← 1
while i ≤ n:
   [ℓ, r] ← [x_i, x_i + 1]
   R ← R ⋃ {[ℓ, r]}
   i ← i + 1
   while i ≤ n and x_i ≤ r:
      i ← i + 1
return R
```

Lemma

The running time of `PointsCoverSorted` is $O(n)$.

Proof

- i changes from 1 to n
- For each i, at most 1 new segment
- Overall, running time is $O(n)$ $\square$

Total Running Time

- `PointsCoverSorted` works in $O(n)$ time
- Sort $\{x_1, x_2, \ldots, x_n\}$, then call `PointsCoverSorted`
- Soon you'll learn to sort in $O(n \log n)$
- Sort + `PointsCoverSorted` is $O(n \log n)$

Asymptotics

- Straightforward solution is $\Omega(2^n)$
- Very long for $n = 50$
- Sort + greedy is $O(n \log n)$
- Fast for $n = 10\,000\,000$
- Huge improvement!

Conclusion

- Straightforward solution is exponential
- Important to reformulate the problem in mathematical terms
- Safe move is to cover leftmost point
- Sort in $O(n \log n)$ + greedy in $O(n)$