MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

Allan Grønlund Jørgensen
University of Aarhus

Locating Interesting Subsequences

Motivation

Finding *interesting* parts of sequences is a problem appearing repeatedly in data analysis.
Locating G+C rich regions of DNA sequences or finding tandem repeats in chromosome data are such problems.
In addition to Bioinformatics, similar and/or identical problems appear in Pattern Matching, Image Processing, and Data Mining.

Problem definitions

Given a Sequence $A[1,\dots,n]$ of Numbers

- Find a subsequence $A[i,\dots,j]$ maximizing $\Sigma^j_{t=i} A[t]$.

Given a Sequence $A[1,\dots,n]$ of Numbers and Integer k

- Find k subsequences maximizing $\Sigma^j_{t=i} A[t]$.
- Find a k'th largest subsequence.

Given a Sequence $A[1,\dots,n]$ of Numbers, Integers l, u and k

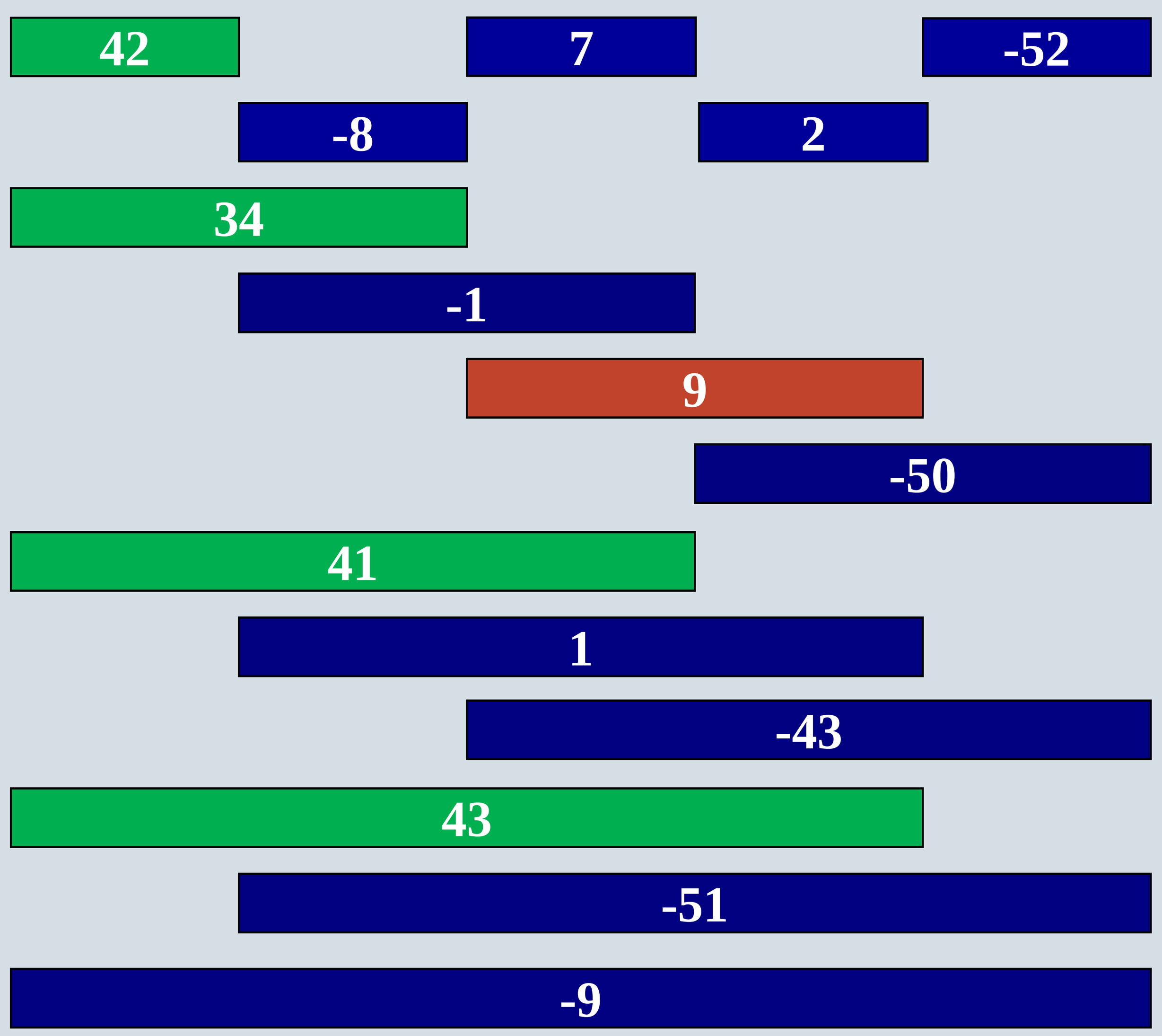
- Find k subsequences maximizing $\Sigma^j_{t=i} A[t]$ among all subsequences of length at least l and at most u.
- Find a k'th largest subsequence among all subsequences of length at least l and at most u.

Problem instance (k=5)

Input Sequence A:

42	-8	7	2	-52

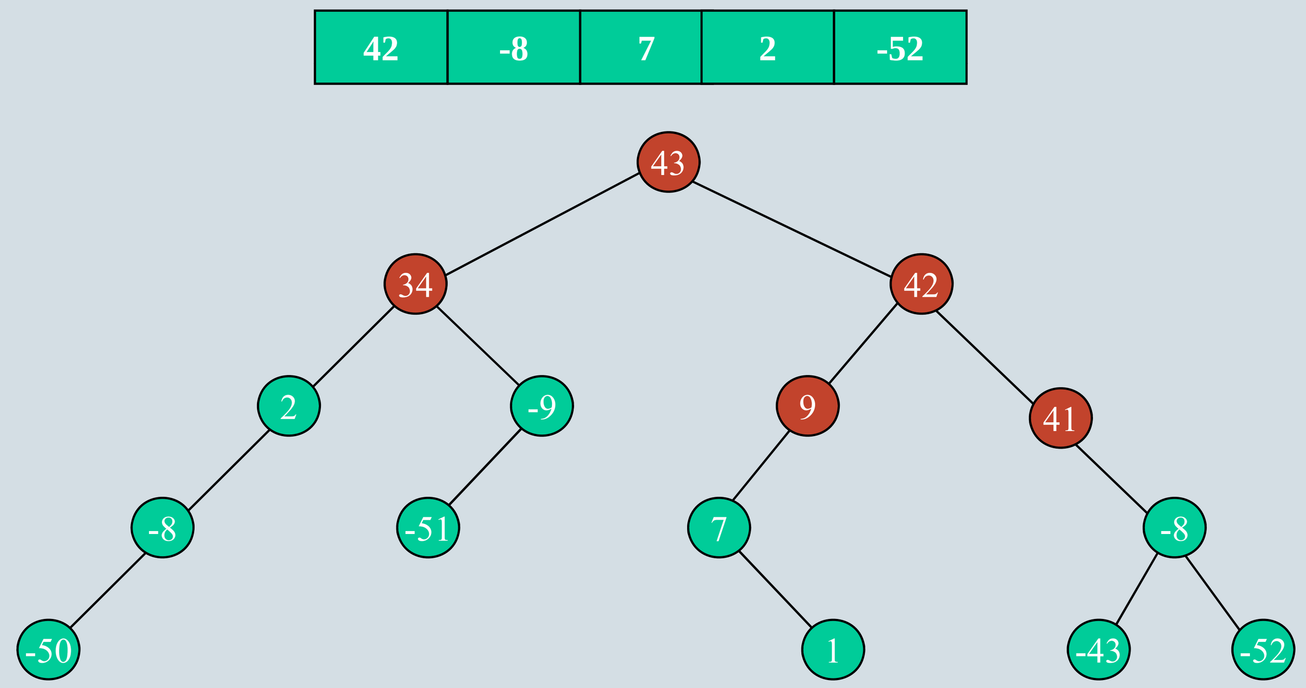
Subsequences $A[i,\dots,j]$:



The k-1 largest are green, the k'th largest is red.

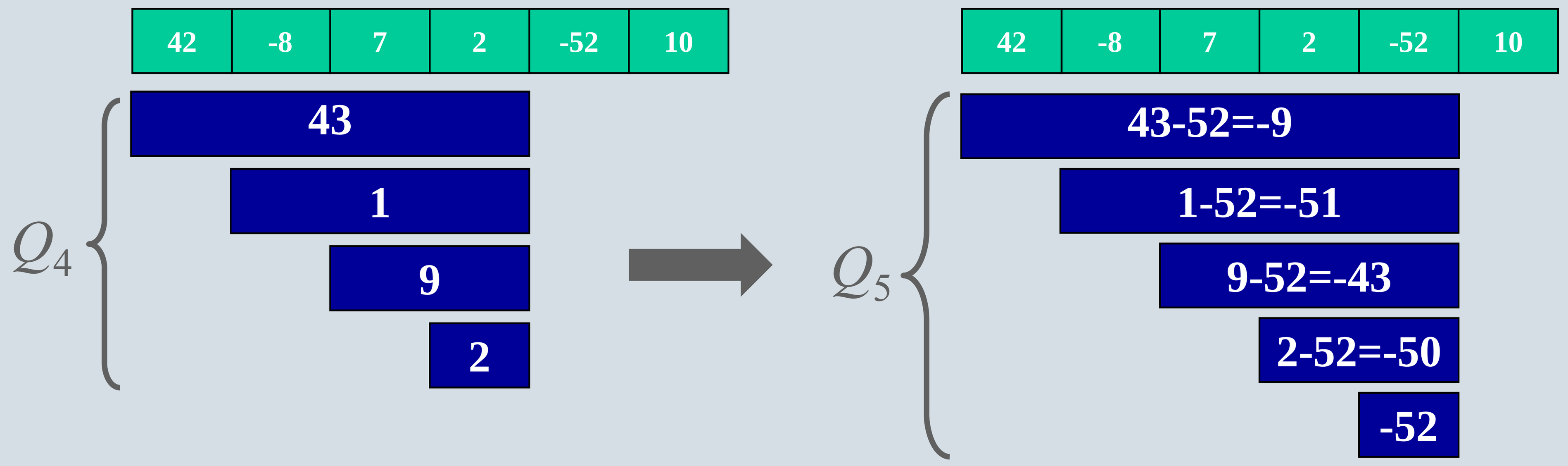
Locating the k largest subsequences: Main ideas

Insert all $n(n+1)/2$ sums in a heap ordered binary tree (values increase towards root).
Find the k largest using Frederickson's heap selection algorithm.

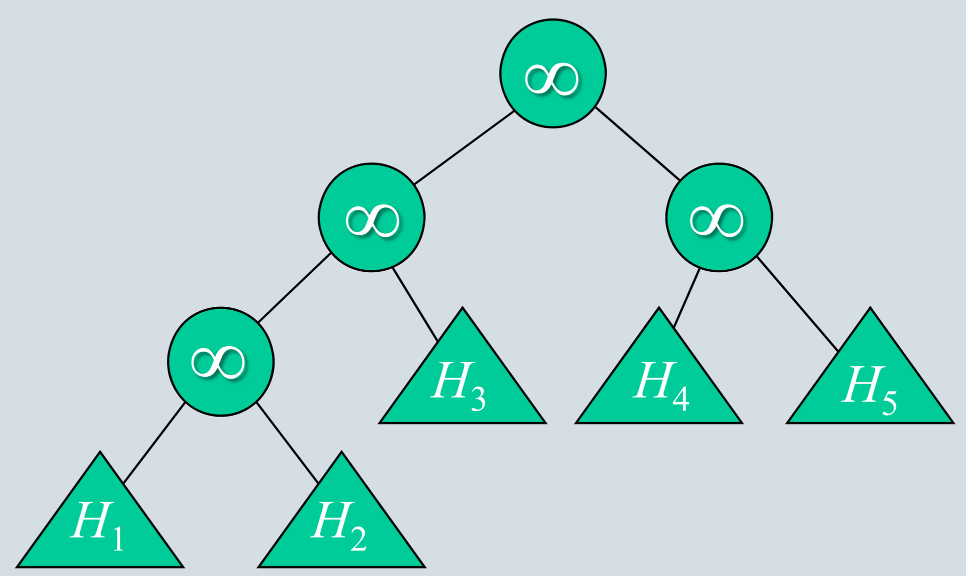


Frederickson's algorithm finds the red nodes in $O(k)$ time (no particular order)

Represent the $O(n^2)$ sums implicitly in a heap ordered binary tree using $O(n)$ space.



A representation of Q_i can be build by adding the same value to all elements in Q_{i-1} and adding one new element. This allows for efficient construction of each set.
Represent each set Q_i using a heap ordered binary tree H_i and join them.



Use Frederickson's algorithm and find the k+n-1 largest and discard the ∞ values.

Results

- An optimal algorithm finding the subsequence $A[i,\dots,j]$ maximizing $\Sigma^j_{t=i} A[t]$ in $O(n)$ time is described in [1].
- In [2] we design an optimal $O(n+k)$ time algorithm using $O(k)$ space.
  This algorithm can be used to solve the 2-dimensional version of the problem in $O(n^3+k)$ time using $O(n+k)$ space and in general the d-dimensional problem in $O(n^{2d-1}+k)$ time and $O(n^{d-1}+k)$ space.
- In [3] we generalize this algorithm to find the subsequences inducing the k largest sums among all subsequences of length between l and u.
- In [3] we show an optimal $O(n \log k/n)$ bound for selecting the subsequence inducing the k'th largest sum, by providing an algorithm with this running time and by proving a matching lower bound.
- We also generalize the selection algorithm to select the k'th largest subsequence among all subsequences of length between l and u, in $O(n \log k/n)$ time.
  Remark that in this case $k \leq (u-l+1)n$.

Bibliography

[1] Bentley, J.: *Programming Pearls: algorithm design techniques*. Commun. ACM 27(9)(1984) 865-873.

[2] Brodal, G. S. and Jørgensen, A. G.: *A linear time algorithm for the k maximal sums problem*. Proc. 32nd International Symposium on Mathematical Foundations of Computer Science.

[3] Brodal, G. S. and Jørgensen, A. G.: *Sum selection in arrays*. In submission.

MADALGO – Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation