Maximum subarray problem:

2	-3	2	1	-1	3	-3	1	-2	1	2	-3	1

Given an array A of length n, consider $A[s, t] = \Sigma_{s \leqq j \leqq t} A[j]$.

Problem: Find the indices s and t maximizing A[s,t]

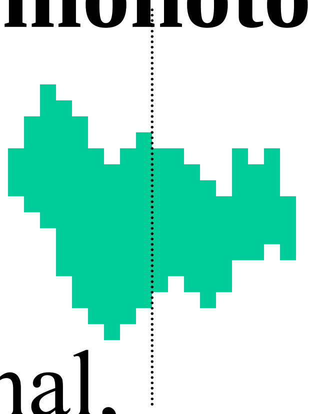
Maximum weight region

Maximum weight region problem: Given a function f*(p) on G, find the region R in the region family **F** maximizing f* (R)

Easy to solve if **F** is the family of

- x-monotone regions

X-monoton



Really easy ?? If you are a professional, you should find a professional solution.

Magic of Algorithms is

the heart of programming and system design

Programming Pearls : Famous column in the magazine CACM (Communications of ACM),

author: John Bentley

Bentley's forewards

I am pleased to announce the inauguration of *Programming Pearls*, a new department of Communications devoted to the seemingly small things that distinguish great programs from other programs.

Programming Pearls，1984，9月

“Algorithm Design Techniques”

Find the interval maximizing the sum of entries in an array

Maximum subarray problem：

2	-3	2	1	-1	3	-3	1	-2	1	2	-3	1

The story (1977)

2	-3	2	1	-1	3	-3	1	-2	1	2	-3	1

- U. Grenander (Reseacher in computer graphics)
  - Find the rectangular region maximizing the entry sum
  - Start with one-dimensional problem
    - Tip: Simplify the problem
  - Naïve method $O(n^3)$
    - n^2 intervals, each interval has n entries
  - Grenander solved in $O(n^2)$ time
  - Ask theory seminar of Bentley for a better method
    - Tip : Talk with people with other expertise

Michael Shamos thought overnight, and

$A(I)$: entry sum of an interval I

Define $P(J)=\mathrm{Max}_{I\subseteq J}\, A(I)$

We can compute the following

$L([s,t])=\mathrm{Max}_{\ j\in[s,t]}A([1,j])$,

$R([s,t])=\mathrm{Max}_{\ j\in[s,t]}A([j,t])$

Divide the inteval J into J1 and J2 at the middle, then

$P(J) = \max\{ P(J_1), P(J_2), R(J_1)+L(J_2)\}$

2	-3	2	1	-1	3	0	1	-2	1	2	-3	1

O(n log n) time

Jay Kadane's DP Algorithm

I (max, k) : Max interval to the left of k

2	-3	2	1	-1	3	-3	1	-2	1	2	-3	1

k

J (max, k) : Max interval whose right end is k

2	-3	2	1	-1	3	-3	1	-2	1	2	-3	1

k

Lemma:

- $I(\max, k) = \text{Max } \{I(\max, k\text{-}1)\ ,\ J(\max, k)\}$
- $J(\max, k) = \text{Max } \{ J(\max, k\text{-}1) + A(k)\ ,\ A(k)\}$

Dynamic Programming procedure runs in linear time

This is not the end of story, just the start

Bentley's open problem : "Solve 2-d problem efficiently"

- $O(n^3)$ (Answers by readers in the next issue of CACM)
  - Steve Mahaney, E.W. Dijkstra ,etc 14 solvers
- Better one? Open Problem

Challenge this problem??

- Not solved in Bentley's seminar ?
- No progress for 15 years, but still ?
- You need idea, timing, and luck

- Tip : If you are interested, think seriously for a week

One day, you may find a solution (1998, Tamaki and T.)

One day, suddenly you find your luck.

M: Could I discuss on my research topic?

T: Of course, I am happy to listen

M: I want to design data mining system of numeric data base, and badly need efficient algorithms?

T: What? What is “data mining” ?

M: This is my problem…..

T: Yes, this is similar to Bentley’s problem. Well, I can solve it, and give details by tomorrow.

Actor : Dr. Morishita, working on database applications

Tip, You should happily accept requests of your friends, and answer as quickly as possible.

Numeric Association Rule

- $100 > t[BP] > 50 \rightarrow$ Diseased = no
- $100 < t[HEIGHT] - t[WEIGHT] < 130 \rightarrow$ Diseased = no

conditional attributes target attribute

AGE	*BP*	*HEIGHT*	*WEIGHT*	*...*	*Diseased*
23	140	168	79	...	Yes
21	91	176	61	...	No
42	129	165	80	...	Yes
30	98	182	57	...	No
...	...	...	...	...	...

Association Rules for Numeric Data

Rules that have the form: $(A \in [x_1, x_2]) \Rightarrow B$

Input: Data base with two attributes A and B.
Attribute A is numeric and B is binary (yes or no)

Output: A *range* I = $[x_1, x_2]$ of attribute value of A

Support : The ratio of data whose A-value is in I

Confidence: Probability that B= 1 under the condition that A is in I

Optimized Numeric Association Rules

$$(A \in [x_1, x_2]) \Rightarrow B$$

We call a rule

- *Well-supported* if support $\geq minsup$ threshold.
- *Confident* if confidence $\geq minconf$ threshold.

Two types of optimized rules:

- Optimized confidence rule : a well-supported rules that maximizes the confidence.
- Optimized support rule : a confident rule that maximizes the support.

Optimized support rule

Maximize A(I) under the condition (A^B)(I)>αA(I)

A	20	32	32	32	34	32	34	32	32	32	32	32	30
A^B	12	14	15	11	21	19	16	15	15	17	13	18	8
C=A^B-0.5A	2	-2	- 1	-5	4	3	-1	-1	-1	1	-3	2	-7

Computed in linear time. How to do it???

Geometric view is helpful.

Optimal confidence rule

A	21	32	23	31	34	32	33	31	22	31	32	33	21
A^B	12	14	20	11	11	9	11	31	12	27	22	23	8

Condition: $A(I) > \beta$ (min-sup)

Maximize $A \wedge B(I)/A(I)$

Use convex hull algorithm

Surprising, isn't it?

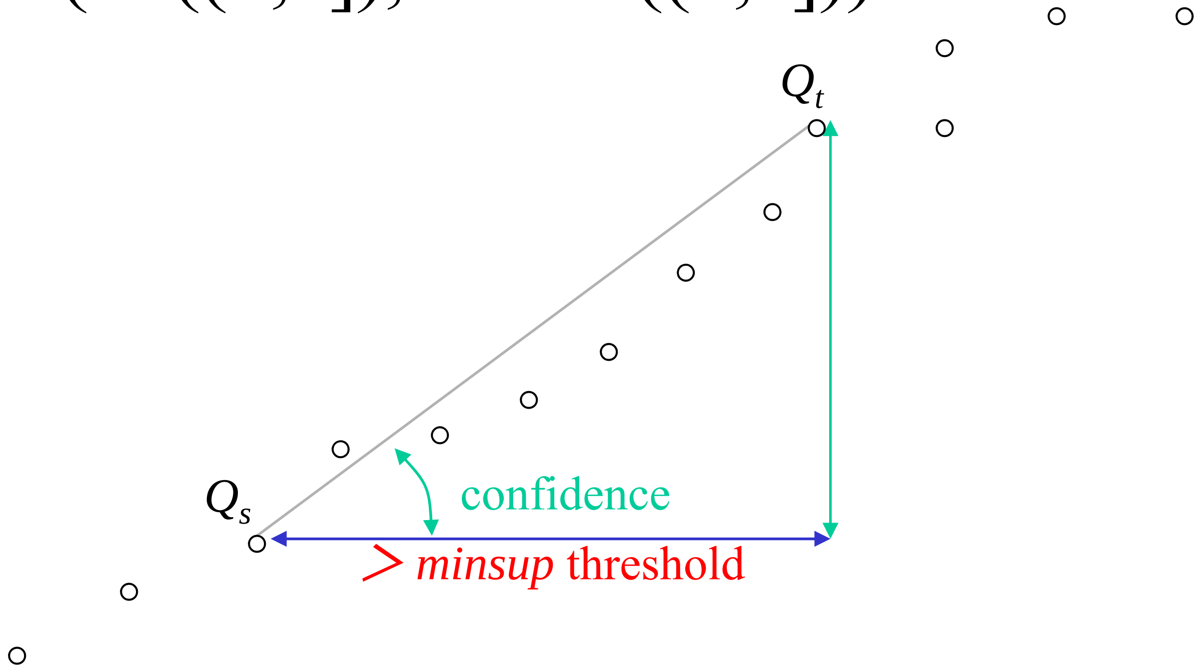
Transformation into a geometric problem

A	21	32	23	31	34	32	33	31	22	31	32	33	21
A^B	12	14	20	11	11	9	11	31	12	27	22	23	8

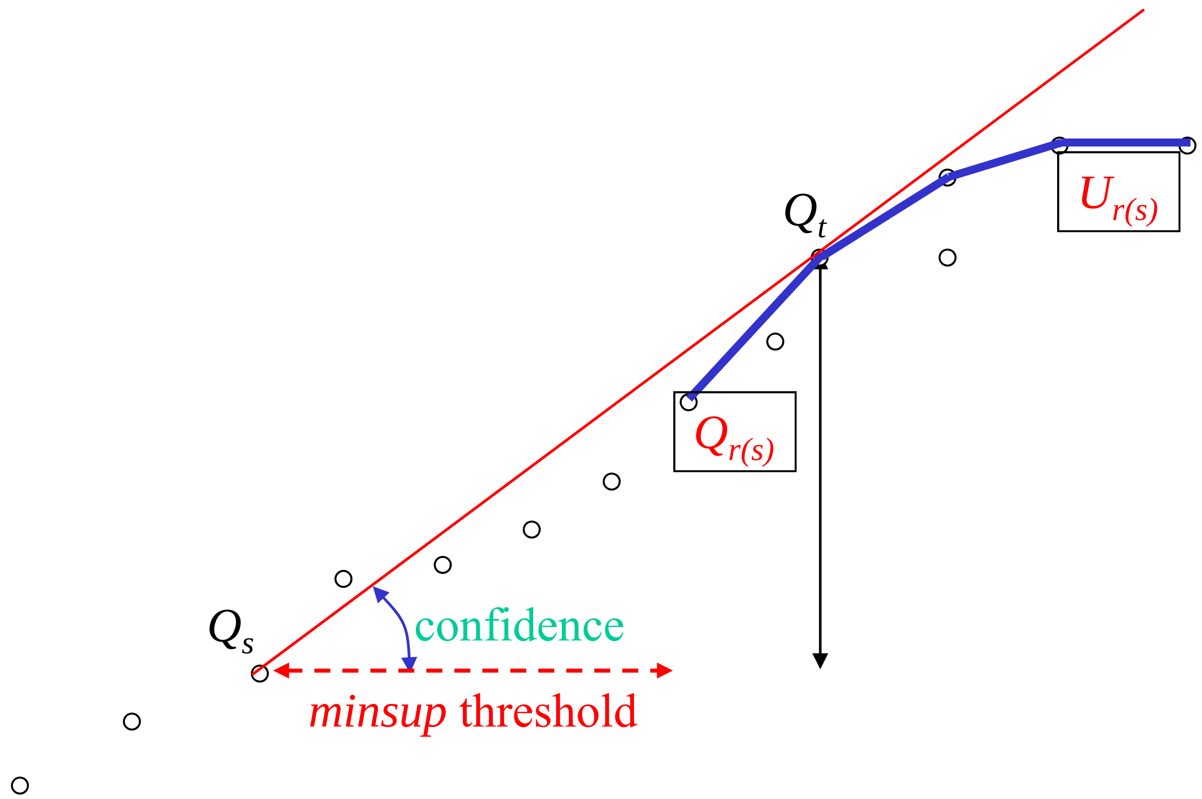
- Generation of sequences of points

$$Q_m = (A([1,m], A^\wedge B([1,m]))$$ Prefix sums

$$Q_s - Q_t = (\mathrm{A}\,((s,t]),\ A^\wedge B\,((s,t]))$$



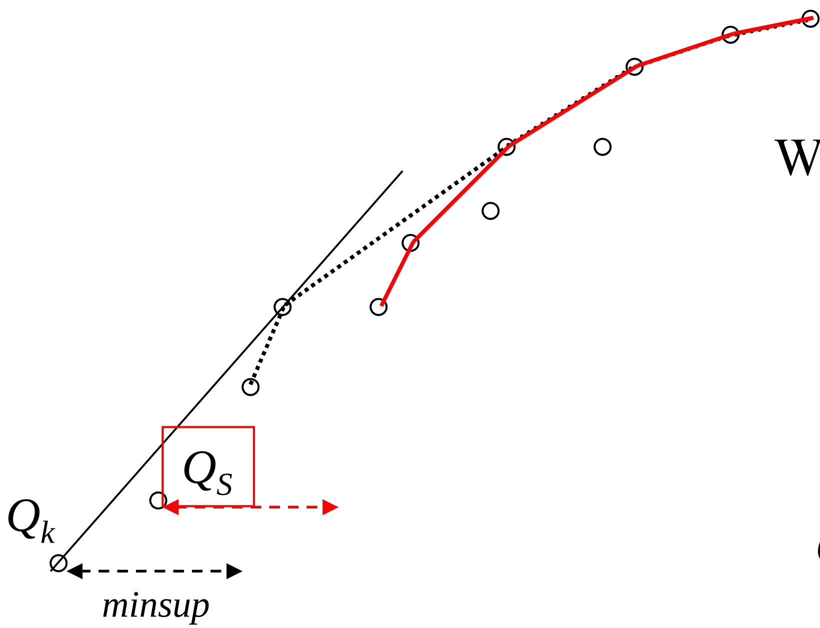
Convex hull again!



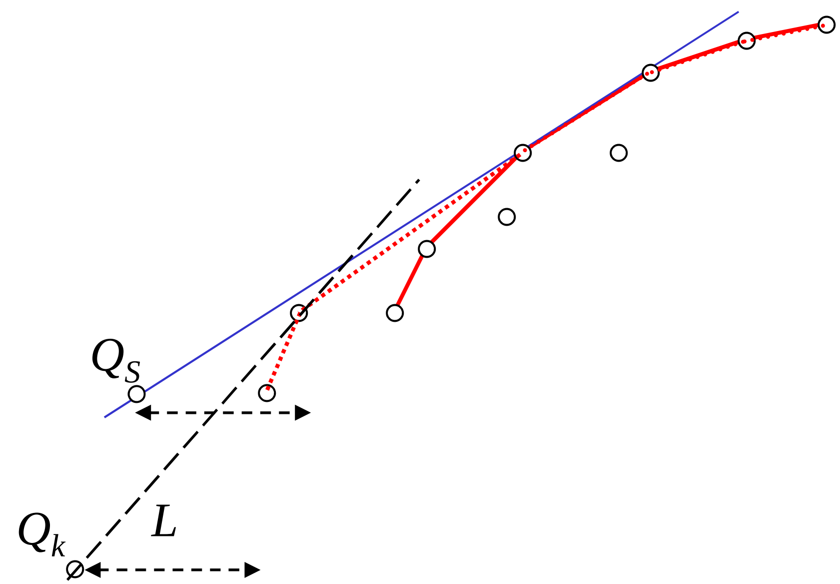
The tangent of Q_s and the corresponding *upper convex hull* for some *s* gives the range of the optimized confidence rule.

Computing Tangents (1)

Compute tangents from left to right.



When Q_s is above the previous tangent L:

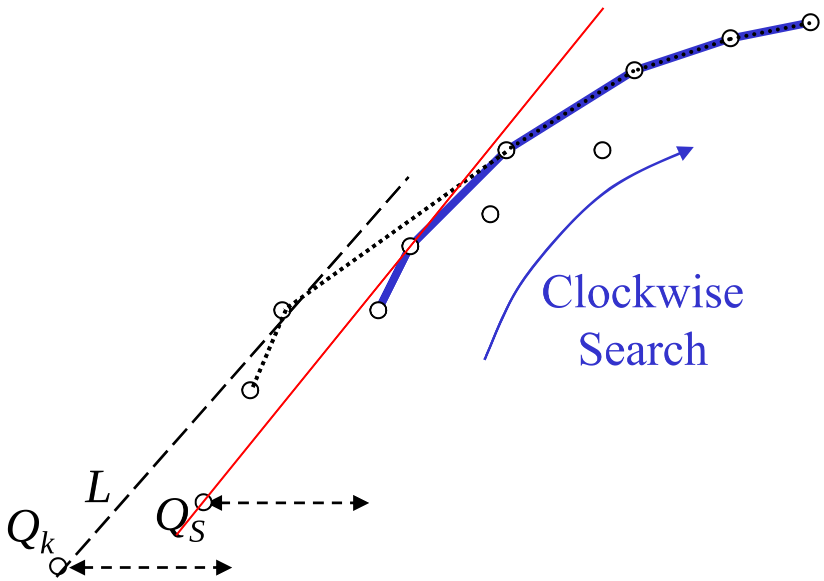


The tangent of Q_s cannot have larger slope than L.
Leave L untouched.

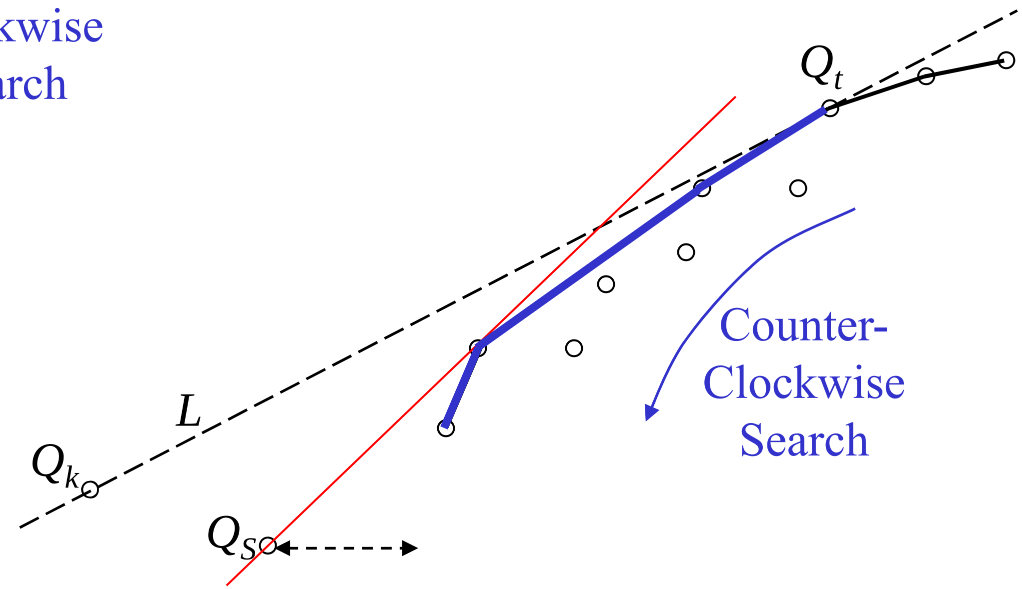
Computing Tangents (2)

When Q_s is below the previous tangent L.

When L does not touch the convex hull:



When L still touches the convex hull:



Algorithm

- Compute the Convex Hull Tree
- Compute tangent lines, moving the anchor point from left to right.
    - Update the tangent points
    - Walk on Convex Hull Tree
- Linear time algorithm