“它不是不灵啦。您没明白，我说十万块钱哪，您是应当买一套。”

“什么叫一套哇？”

“一套。一套是两张：一张打着伞的，一张夹着伞的。下雨的时候，您看这张；不下雨您再看那张啊！”

01-XD

Convex Hull

Chan's Algorithm

Junhui DENG

deng@tsinghua.edu.cn

Output Sensitivity

- The $\boxed{\log n}$ factor of the CH algorithm complexity arises from the fact that you need to $\boxed{\text{sort}}$ the up to $\boxed{n}$ points on the hull
- Suppose that you were told (e.g., by an $\boxed{\text{oracle}}$) that there would be only $\boxed{h}$ points on the hull and $\boxed{h \ll n}$
- Then the reasonable running time should be $\boxed{\mathcal{O}(n\log \boxed{h})}$ instead of $\boxed{\mathcal{O}(n\log \boxed{n})}$

  // We will see later that $\boxed{\mathcal{O}(n\log \boxed{h})}$ is $\boxed{\text{optimal}}$
- Possible to get such an $\boxed{\text{output sensitive}}$ CH algorithm? How to?

The Ultimate Algorithm

- Based on a clever pruning method,

  Kirkpatrick & Seidel discovered an $\mathcal{O}(n\log \boxed{h})$-time algorithm in 1986

  //D. G. Kirkpatrick & R. Seidel, The Ultimate Planar Convex Hull Algorithm?

  //SIAM Journal on Computing, vol. 15, No. 1, February 1986, pp. 287-299

  The convex hull of a set of n points in the plane can be constructed

  in $\boxed{\mathcal{O}(n\log \boxed{h})}$ time, where $\boxed{h}$ is the number of points on the hull

- However, this algorithm is relatively complicated and

  is almost impossible for practical implementation

- Though, this problem had been considered closed

  until around 10 years later when T. M. Chan came up with

  a much simpler algorithm with the same running time

Graham Scan + Jarvis March

❖ By combining two slower algorithms together,
Chan got an algorithm that is faster than either one

1) The problem with $\boxed{\text{GS}}$ is that it
has to $\boxed{\text{sort}}$ first all the points, and hence
is doomed to have an $\boxed{\Omega(n\log\boxed{n})}$ running time,
irrespective of the hull size

2) On the other hand, if you have $\boxed{\text{few}}$ vertices on the hull,
$\boxed{\text{JM}}$ can perform better but
it takes $\boxed{\Omega(n)}$ time for each EE

Chan's Idea

1) Partition the points into $\boxed{r}$ groups of equal size $\boxed{m}$

$$\boxed{r = \lceil n/m \rceil}$$

2) For each group, construct its hull using $\boxed{\text{GS}}$

Each subhull costs

$$\boxed{\mathcal{O}(m * \log m)}$$

time, and hence subhulls for all groups can be obtained in time of

$$\boxed{r} \times \boxed{\mathcal{O}(m * \log m)} = \boxed{\mathcal{O}(n * \log m)}$$

❖ How to $\boxed{\text{merge}}$ the $\boxed{r}$ subhulls? //And more important ...

How fast can it be done?

Merging Subhulls by Jarvis March

❖ During $\boxed{\text{JM}}$, we take advantage of the fact that

the tangent between a point and a subhull of size $\boxed{m}$ can be found

in $\boxed{\mathcal{O}(\log m)}$ time

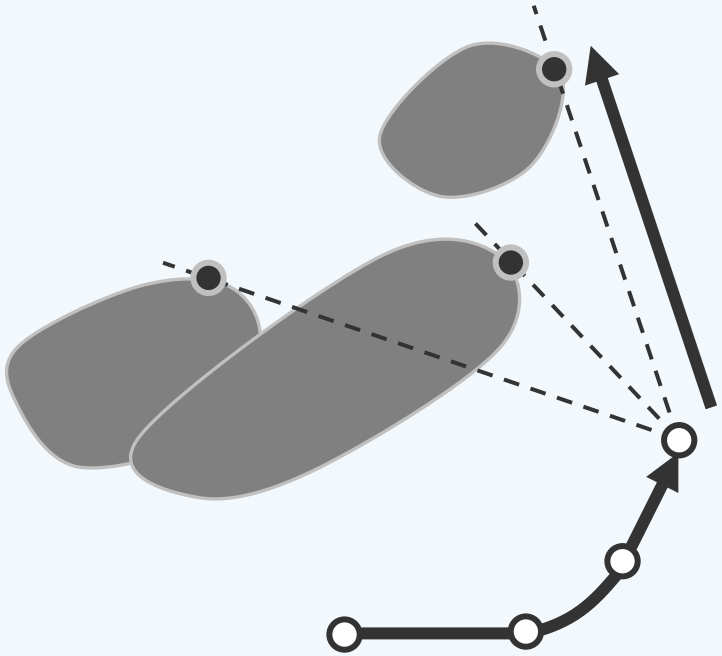
❖ To achieve this, for example, we can

1) store the vertices of the subhull

   in a $\boxed{\text{linear array}}$ and

2) compute the tangent

   in a $\boxed{\text{binary search}}$ manner



❖ Hence during each of the $\boxed{h}$ iterations

1) $\boxed{r}$ tangents can be found in $\boxed{\mathcal{O}(r * \log m)}$ time and

2) an EE can be found in $\boxed{\mathcal{O}(r)}$ additional time

Complexity

- As a whole, the final hull can be constructed from r subhulls in time of

  $h \times \mathcal{O}(r * \log m) = \mathcal{O}(h * r * \log m) = \mathcal{O}(\boxed{\boxed{hn/m} \times \log m})$

- Combining the 2 steps ($\boxed{GS + JM}$) above, we get that

  for any $1 < m < n$, the convex hull of n points in the plane

  can be constructed in $\mathcal{O}(\boxed{\boxed{(n + hn/m)} \times \log m})$ time

- You might have noticed that one thing is ignored here ...

- Before running JM, can we find the $\boxed{\text{first}}$ vertex on the final hull

  in, say, $\boxed{\mathcal{O}(\log m)}$ time?

- Convince yourself that, although the answer is $\boxed{\text{no}}$,

  it does $\boxed{\text{not}}$ affect the complexity of the algorithm as a whole

$\boxed{(n + hn/m)} \times \log m$

1) If $\boxed{m = n}$, the total running time will be $\boxed{\mathcal{O}(n\log n)}$

It is not surprising, because we were actually running GS on a $\boxed{\text{single}}$ group

2) If $\boxed{m = 2}$, the total running time will be $\boxed{\mathcal{O}(nh)}$

It doesn't surprise us either, since, in fact

we're running JM on n/2 groups, each of which consists of only 2 points

3) If $\boxed{m = h}$, the total running time will be $\boxed{\mathcal{O}(n\log h)}$! //just as expected

❖ But, the key problem here is that

how could we $\boxed{\text{know } \boxed{h} \text{ in advance}}$ so that

we can choose an $\boxed{m = \Theta(h)}$ $\boxed{\text{before}}$ starting the algorithm?

//can an oracle help here?

Partial Convex Hull

❖ PartialHull (P, n, m) //m = #points in each group

Partition the n points in P into $\boxed{r = \lceil n/m \rceil}$ groups

Compute $\boxed{r}$ subhulls using $\boxed{\text{GS}}$ and

store the vertices of each subhull in an ordered array reps.

Run $\boxed{\text{JM}}$ on the $\boxed{r}$ subhulls for $\boxed{\boxed{\text{no more than m}}\text{ steps}}$

Once the hull becomes $\boxed{\text{closed}}$, stop march and return successfully

If the hull doesn't become closed yet after $\boxed{m}$ steps we

know that $\boxed{m}$ is $\boxed{\text{too small}}$ and

may try a $\boxed{\text{bigger}}$ m later //next m = ? & later = when?

Partial Convex Hull

1) Chan's partial convex hull algorithm constructs the hull as long as

   $\boxed{m \geq h}$

2) If $\boxed{m < h}$, it returns a special error status $\boxed{\text{M\_NOT\_BIG\_ENOUGH}}$

3) In both cases, the algorithm will terminate after $\boxed{\mathcal{O}(m)}$ iterations of $\boxed{\text{JM}}$

❖ Therefore,

each call (with an argument m) for Chan's partial CH algorithm

will return in time of

$$\mathcal{O}(n\log m) + \mathcal{O}(m) * \mathcal{O}(r\log m) = \boxed{\mathcal{O}(n\log m)}$$

Guessing m

- Just as stated above, the key problem here is that how could we call the partial CH algorithm with an appropriate $\boxed{m}$?
- Chan suggests that
  1) we start with a $\boxed{\text{small m}}$, and
  2) each time the partial CH algorithm returns failure, $\boxed{\text{increase m}}$ before trying it again
- $\boxed{\text{How to}}$ increase m?
- $\boxed{\text{How much}}$ should it be increased each time?

Exponential Search

- Sequential search: keep trying with $\boxed{m = \boxed{2, 3, 4, 5, \ldots}}$, until $m \geq h$

  $n \times (\log 2 + \log 3 + \log 4 + \ldots + \log h) = n * \boxed{\log(h!)} = \mathcal{O}(n * \boxed{h\log h})$

- Binary search: $\boxed{m = \boxed{2, 4, 8, 16, \ldots}}$, until $m \geq h$ //arithmetic progression

  $n \times (\log 2 + \log 4 + \log 8 + \ldots + \log h) = \boxed{n * \log^2 h}$

- Exponential search: $\boxed{m_i = m_{i-1}^c, \boxed{i = 1, 2, 3, \ldots}}$, until $m \geq h$

  For example, if constant $\boxed{c = 2}$, then $\boxed{m = \boxed{2, 4, 16, 256, \ldots}}$

  $n \times (\log 2 + \log 4 + \log 16 + \log 256 + \ldots + \log h)$ //geometric progression

  $n \times (1 + 2 + 4 + 8 + \ldots + \log h) = \boxed{n * \log h}$

Lower Bound

- We'll first give such a lower bound

  on the following "simpler" decision problem

- Convex Hull Size Verification

  Given a point set P and an integer h, does CH(P) have h distinct vertices?

- Based on the algebra decision tree model, we can prove that

  CHSV requires $\Omega(n\log h)$ time to solve

- This problem is not harder than the convex hull problem because

  $\boxed{\text{CHSV} \leq_N \text{CH}}$

- Hence, $\Omega(n\log h)$ is also an lower bound for CH and

  Chan's algorithm is optimal