Lecture 20: Peak Finding in 2D

COMS10007 - Algorithms

Dr. Christian Konrad

30.04.2019

Peak Finding

Let $A = a_0, a_1, \ldots, a_{n-1}$ be an array of integers of length n

0	1	2	3	4	5	6	7	8	9
a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9

Definition: (Peak)
Integer a_i is a *peak* if adjacent integers are not larger than a_i

Example:

4	3	9	10	14	8	7	2	2	2

Peak Finding

Let $A = a_0, a_1, \ldots, a_{n-1}$ be an array of integers of length n

0	1	2	3	4	5	6	7	8	9
a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9

Definition: (Peak)
Integer a_i is a *peak* if adjacent integers are not larger than a_i

Example:

4	3	9	10	14	8	7	2	2	2

Peak Finding

Let A be an n-by-m matrix of integers

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1m} \\ A_{21} & \ddots & & & \\ A_{31} & & \ddots & & \\ \vdots & & & \ddots & \\ A_{n1} & A_{n2} & A_{n3} & \dots & A_{nm} \end{pmatrix}$$

Definition: (Peak in 2D)
Integer A_{ij} is a *peak* if adjacent integers are not larger than A_{ij}

Example and Trivial Algorithm

How many peaks are contained in this matrix?

$$\begin{pmatrix} 1 & 5 & 8 & 3 \\ 2 & 1 & 8 & 9 \\ 3 & 1 & 1 & 2 \\ 7 & 7 & 8 & 10 \\ 2 & 1 & 1 & 1 \end{pmatrix}$$

Trivial Algorithm

- For each position i, j, check whether $A_{i,j}$ is a peak
- There are $n \cdot m$ positions
- Checking whether $A_{i,j}$ is a peak takes time $O(1)$
- Runtime: $O(nm)$

Example and Trivial Algorithm

How many peaks are contained in this matrix?

$$\begin{pmatrix} 1 & 5 & \color{red}{8} & 3 \\ 2 & 1 & 8 & \color{red}{9} \\ 3 & 1 & 1 & 2 \\ \color{red}{7} & 7 & 8 & \color{red}{10} \\ 2 & 1 & 1 & 1 \end{pmatrix}$$

Trivial Algorithm

- For each position i, j, check whether $A_{i,j}$ is a peak
- There are $n \cdot m$ positions
- Checking whether $A_{i,j}$ is a peak takes time $O(1)$
- Runtime: $O(nm)$

How can we do better?

Divide-and-Conquer Solution

Divide-and-Conquer

- **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
- **Conquer** the subproblems by solving them recursively. If the subproblems are small enough, just solve them in a straightforward manner.
- **Combine** the solutions to the subproblems into the solution for the original problem.

1D Peak Finding

Divide-and-Conquer in 1D: FAST-PEAK-FINDING

1. Check whether $A[\lfloor n/2 \rfloor]$ is a peak, if yes then return $A[\lfloor n/2 \rfloor]$
2. Else, if $A[\lfloor n/2 \rfloor - 1] > A[\lfloor n/2 \rfloor]$ then recursively find a peak in $A[0, \lfloor n/2 \rfloor - 1]$
3. Else, recursively find a peak in $A[\lfloor n/2 \rfloor + 1, n - 1]$

Crucial Property

Crucial Property:
When recursing on subarray, need to make sure that peak in subarray is also peak in initial array

Example:

$$A = 1 \quad 2 \quad 3 \quad \mathbf{4} \quad {\color{red}5} \quad {\color{red}6} \quad {\color{red}7} \quad {\color{red}8}$$

- Algorithm first inspects 4 and recurses on right half

$${\color{red}5} \quad {\color{red}6} \quad {\color{red}7} \quad {\color{red}\mathbf{8}}$$

  Will eventually find the only peak **8**
- Suppose we recursed on left half

$$1 \quad 2 \quad \mathbf{3}$$

  peak in 1 2 **3** is not a peak in A!

2D Peak Finding

Divide-and-Conquer: Divide step

- Find maximum among central column and boundary
- If it is not a peak, conquer either on left or right half

$$\begin{pmatrix} A_{11} & \dots & A_{1,\frac{m}{2}-1} & A_{1,\frac{m}{2}} & A_{1,\frac{m}{2}+1} & \dots & A_{1m} \\ A_{21} & \dots & A_{2,\frac{m}{2}-1} & A_{2,\frac{m}{2}} & A_{2,\frac{m}{2}+1} & \dots & A_{2m} \\ \vdots & & & \vdots & & & \vdots \\ A_{n-1,1} & \dots & A_{n-1,\frac{m}{2}-1} & A_{n-1,\frac{m}{2}} & A_{n-1,\frac{m}{2}+1} & \dots & A_{n-1,m} \\ A_{n,1} & \dots & A_{n,\frac{m}{2}-1} & A_{n,\frac{m}{2}} & A_{n,\frac{m}{2}+1} & \dots & A_{n,m} \end{pmatrix}$$

- In each recursive call, number of elements in matrix halves (at least)
- Hence $O(\log(mn))$ recursive calls
- In each call: $O(n + m)$, thus in total $O((n + m)\log(nm))$
- Can be improved to $O(\max\{m, n\})$!

2D Peak Finding

Divide-and-Conquer: Divide step

- Find maximum among central column and boundary
- If it is not a peak, conquer either on left or right half

$$\begin{pmatrix} A_{11} & \dots & A_{1,\frac{m}{2}-1} \\ A_{21} & \dots & A_{2,\frac{m}{2}-1} \\ \vdots & & \\ A_{n-1,1} & \dots & A_{n-1,\frac{m}{2}-1} \\ A_{n,1} & \dots & A_{n,\frac{m}{2}-1} \end{pmatrix}$$

- In each recursive call, number of elements in matrix halves (at least)
- Hence $O(\log(mn))$ recursive calls
- In each call: $O(n+m)$, thus in total $O((n+m)\log(nm))$
- Can be improved to $O(\max\{m,n\})$!

Recursion on which side?

- Since maximum not a peak, a not considered neighbor larger
- Recurse on the side that contains this larger neighbor

$$\begin{pmatrix} A_{11} & \dots & A_{1,\frac{m}{2}-1} & A_{1,\frac{m}{2}} & A_{1,\frac{m}{2}+1} & \dots & A_{1m} \\ A_{21} & \dots & A_{2,\frac{m}{2}-1} & A_{2,\frac{m}{2}} & A_{2,\frac{m}{2}+1} & \dots & A_{2m} \\ \vdots & & & \vdots & & & \vdots \\ A_{n-1,1} & \dots & A_{n-1,\frac{m}{2}-1} & A_{n-1,\frac{m}{2}} & A_{n-1,\frac{m}{2}+1} & \dots & A_{n-1,m} \\ A_{n,1} & \dots & A_{n,\frac{m}{2}-1} & A_{n,\frac{m}{2}} & A_{n,\frac{m}{2}+1} & \dots & A_{n,m} \end{pmatrix}$$

Recursion on which side?

- Since maximum not a peak, a not considered neighbor larger
- Recurse on the side that contains this larger neighbor

$$\begin{pmatrix} A_{11} & \dots & A_{1,\frac{m}{2}-1} & A_{1,\frac{m}{2}} & A_{1,\frac{m}{2}+1} & \dots & A_{1m} \\ A_{21} & \dots & A_{2,\frac{m}{2}-1} & A_{2,\frac{m}{2}} & A_{2,\frac{m}{2}+1} & \dots & A_{2m} \\ \vdots & & & \vdots & & & \vdots \\ A_{n-1,1} & \dots & A_{n-1,\frac{m}{2}-1} & A_{n-1,\frac{m}{2}} & A_{n-1,\frac{m}{2}+1} & \dots & A_{n-1,m} \\ A_{n,1} & \dots & A_{n,\frac{m}{2}-1} & A_{n,\frac{m}{2}} & \mathbf{A_{n,\frac{m}{2}+1}} & \dots & A_{n,m} \end{pmatrix}$$

Recursion on which side?

- Since maximum not a peak, a not considered neighbor larger
- Recurse on the side that contains this larger neighbor

$$\begin{pmatrix} A_{11} & \dots & A_{1,\frac{m}{2}-1} & A_{1,\frac{m}{2}} & A_{1,\frac{m}{2}+1} & \dots & A_{1m} \\ A_{21} & \dots & A_{2,\frac{m}{2}-1} & \mathbf{A_{2,\frac{m}{2}}} & A_{2,\frac{m}{2}+1} & \dots & A_{2m} \\ \vdots & & & \vdots & & & \vdots \\ A_{n-1,1} & \dots & A_{n-1,\frac{m}{2}-1} & A_{n-1,\frac{m}{2}} & A_{n-1,\frac{m}{2}+1} & \dots & A_{n-1,m} \\ A_{n,1} & \dots & A_{n,\frac{m}{2}-1} & A_{n,\frac{m}{2}} & A_{n,\frac{m}{2}+1} & \dots & A_{n,m} \end{pmatrix}$$

Why Does it Work?

Correctness:

- Suppose algorithm finds peak in a submatrix A'
- Why is this also a peak in A?

First Case: Peak is in central column of A' ✓

Second Case: Peak in bottom or top boundary of A'
Only happens in first iteration ✓

$$\begin{pmatrix} A'_{11} & \dots & A'_{1,\frac{m'}{2}-1} & A'_{1,\frac{m'}{2}} & A'_{1,\frac{m'}{2}+1} & \dots & A'_{1m'} \\ A'_{21} & \dots & A'_{2,\frac{m'}{2}-1} & A'_{2,\frac{m'}{2}} & A'_{2,\frac{m'}{2}+1} & \dots & A'_{2m'} \\ \vdots & & & \vdots & & & \vdots \\ A'_{n'-1,1} & \dots & A'_{n'-1,\frac{m'}{2}-1} & A'_{n'-1,\frac{m'}{2}} & A'_{n'-1,\frac{m'}{2}+1} & \dots & A'_{n'-1,m'} \\ A'_{n',1} & \dots & A'_{n',\frac{m'}{2}-1} & A'_{n',\frac{m'}{2}} & A'_{n',\frac{m'}{2}+1} & \dots & A'_{n',m'} \end{pmatrix}$$

Why Does it Work?

Correctness:

- Suppose algorithm finds peak in a submatrix A'
- Why is this also a peak in A?

First Case: Peak is in central column of A' ✓

Second Case: Peak in bottom or top boundary of A'
Only happens in first iteration ✓

$$\begin{pmatrix} A'_{11} & \dots & A'_{1,\frac{m'}{2}-1} & A'_{1,\frac{m'}{2}} & A'_{1,\frac{m'}{2}+1} & \dots & A'_{1m'} \\ A'_{21} & \dots & A'_{2,\frac{m'}{2}-1} & \mathbf{A'_{2,\frac{m'}{2}}} & A'_{2,\frac{m'}{2}+1} & \dots & A'_{2m'} \\ \vdots & & & \vdots & & & \vdots \\ A'_{n'-1,1} & \dots & A'_{n'-1,\frac{m'}{2}-1} & A'_{n'-1,\frac{m'}{2}} & A'_{n'-1,\frac{m'}{2}+1} & \dots & A'_{n'-1,m'} \\ A'_{n',1} & \dots & A'_{n',\frac{m'}{2}-1} & A'_{n',\frac{m'}{2}} & A'_{n',\frac{m'}{2}+1} & \dots & A'_{n',m'} \end{pmatrix}$$

Why Does it Work?

Correctness:

- Suppose algorithm finds peak in a submatrix A'
- Why is this also a peak in A?

First Case: Peak is in central column of A' ✓

Second Case: Peak in bottom or top boundary of A'
Only happens in first iteration ✓

$$\begin{pmatrix} A'_{11} & \dots & \mathbf{A'_{1,\frac{m'}{2}-1}} & A'_{1,\frac{m'}{2}} & A'_{1,\frac{m'}{2}+1} & \dots & A'_{1m'} \\ A'_{21} & \dots & A'_{2,\frac{m'}{2}-1} & A'_{2,\frac{m'}{2}} & A'_{2,\frac{m'}{2}+1} & \dots & A'_{2m'} \\ \vdots & & & \vdots & & & \vdots \\ A'_{n'-1,1} & \dots & A'_{n'-1,\frac{m'}{2}-1} & A'_{n'-1,\frac{m'}{2}} & A'_{n'-1,\frac{m'}{2}+1} & \dots & A'_{n'-1,m'} \\ A'_{n',1} & \dots & A'_{n',\frac{m'}{2}-1} & A'_{n',\frac{m'}{2}} & A'_{n',\frac{m'}{2}+1} & \dots & A'_{n',m'} \end{pmatrix}$$

Why Does it Work? (2)

Peak in Left or Right Boundary of A':

$$\begin{pmatrix} A'_{11} & \dots & A'_{1,\frac{m'}{2}-1} & A'_{1,\frac{m'}{2}} & A'_{1,\frac{m'}{2}+1} & \dots & A'_{1m'} \\ A'_{21} & \dots & A'_{2,\frac{m'}{2}-1} & A'_{2,\frac{m'}{2}} & A'_{2,\frac{m'}{2}+1} & \dots & A'_{2m'} \\ \vdots & & & \vdots & & & \vdots \\ \mathbf{A'_{n'-1,1}} & \dots & A'_{n'-1,\frac{m'}{2}-1} & A'_{n'-1,\frac{m'}{2}} & A'_{n'-1,\frac{m'}{2}+1} & \dots & A'_{n'-1,m'} \\ A'_{n',1} & \dots & A'_{n',\frac{m'}{2}-1} & A'_{n',\frac{m'}{2}} & A'_{n',\frac{m'}{2}+1} & \dots & A'_{n',m'} \end{pmatrix}$$

- Need to make sure that $A'_{n'-1,m'}$ is not smaller than element left of it in A (if it exists)
- **Observe:** Element left of it is in central column of a matrix that was considered earlier

Key Lemma

Lemma

Let $A = A_1, A_2, A_3, \dots$ be the sequence of matrices considered by the algorithm. Let m_i be the maximum of the central column and the boundary in A_i. Then:

$$m_{i+1} \geq m_i \ .$$

Proof. If m_i is in bottom/top/left/right boundary (excluding the elements that are also in central column) of A_i, then m_i is also in boundary of A_{i+1}. Hence, $m_{i+1} \geq m_i$.
Suppose m_i is in central column. Since it is not a peak, either left or right element is larger. Let this element be x. Hence, $x > m_i$. Observe that x is in boundary of A_{i+1}. Since $m_{i+1} \geq x$, we conclude $m_{i+1} > m_i$. □

$\rightarrow$ Peak found in left or right column in A' is also peak in A! (establishes correctness of algorithm)

Summary

Peak Finding in 2D

- Divide and conquer algorithm
- Finds a peak in time $O((m+n)\log(mn))$ on an n-by-m matrix
- For square (n-by-n) matrices, this is $O(n \log(n^2)) = O(n \log n)$

Improvement (for simplicity suppose that A is an n-by-n matrix)

- If $\#$ columns $\geq$ $\#$ rows then recurse horizontally as before
- If $\#$ columns $<$ $\#$ rows then recurse vertically

Observe:

- Vertical and horizontal splits alternate
- After two recursions we have n'-by-n' matrix with $n' < n/2$

Runtime of Improved Algorithm

Analysis: (sketch)

- In iteration 1, matrix is of size n-by-n
- In iteration 3, matrix is of size at most $n/2$-by-$n/2$
- In iteration 5, matrix is of size at most $n/4$-by-$n/4$
- ...

$$\begin{aligned}
\text{Runtime} &\leq \sum_{i=1}^{\log(n^2)} \text{Runtime in it. } i \leq 2 \cdot \sum_{i=1,3,5,7,\ldots}^{2\log n} \text{Runtime in it. } i \\
&= 2 \cdot \sum_{i=1}^{\log n} \text{Runtime on matrix with dimensions } n/2^{i-1} \times n/2^{i-1} \\
&= 2 \cdot \sum_{i=1}^{\log n} O(n/2^{i-1}) = O(n) \sum_{i=1}^{\log n} O(\frac{1}{2^{i-1}}) = O(n) \cdot O(1) = O(n) \, .
\end{aligned}$$