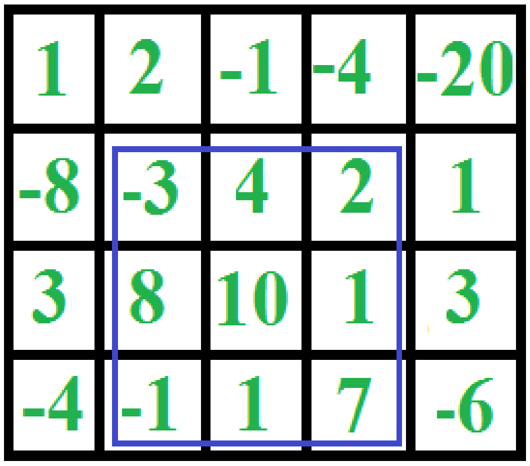
Dynamic Programming | Set 27 (Maximum sum rectangle in a 2D matrix)

Given a 2D array, find the maximum sum subarray in it. For example, in the following 2D array, the maximum sum subarray is highlighted with blue rectangle and sum of this subarray is 29.



1	2	-1	-4	-20
-8	-3	4	2	1
3	8	10	1	3
-4	-1	1	7	-6

This problem is mainly an extension of Largest Sum Contiguous Subarray for 1D array.

The **naive solution** for this problem is to check every possible rectangle in given 2D array. This solution requires 4 nested loops and time complexity of this solution would be O(n^4).

Kadane's algorithm for 1D array can be used to reduce the time complexity to O(n^3). The idea is to fix the left and right columns one by one and find the maximum sum contiguous rows for every left and right column pair. We basically find top and bottom row numbers (which have maximum sum) for every fixed left and right column pair. To find the top and bottom row numbers, calculate sun of elements in every row from left to right and store these sums in an array say temp[]. So temp[i] indicates sum of elements from left to right in row i. If we apply Kadane's 1D algorithm on temp[], and get the maximum sum subarray of temp, this maximum sum would be the maximum possible sum with left and right as boundary columns. To get the overall maximum sum, we compare this sum with the maximum sum so far.

```
// Program to find maximum sum subarray in a given 2D array
#include <stdio.h>
#include <string.h>
#include <limits.h>
#define ROW 4
#define COL 5

// Implementation of Kadane's algorithm for 1D array. The function returns th
// maximum sum and stores starting and ending indexes of the maximum sum suba
// at addresses pointed by start and finish pointers respectively.
int kadane(int* arr, int* start, int* finish, int n)
{
    // initialize sum, maxSum and
    int sum = 0, maxSum = INT_MIN, i;

    // Just some initial value to check for all negative values case
    *finish = -1;

    // local variable
    int local_start = 0;

    for (i = 0; i < n; ++i)
    {
        sum += arr[i];
        if (sum < 0)
        {
            sum = 0;
            local_start = i+1;
        }
        else if (sum > maxSum)
        {
            maxSum = sum;
            *start = local_start;
            *finish = i;
        }
    }

     // There is at-least one non-negative number
    if (*finish != -1)
        return maxSum;

    // Special Case: When all numbers in arr[] are negative
    maxSum = arr[0];
    *start = *finish = 0;

    // Find the maximum element in array
    for (i = 1; i < n; i++)
    {
        if (arr[i] > maxSum)
        {
            maxSum = arr[i];
            *start = *finish = i;
        }
    }
    return maxSum;
}

// The main function that finds maximum sum rectangle in M[][]
void findMaxSum(int M[][COL])
{
    // Variables to store the final output
    int maxSum = INT_MIN, finalLeft, finalRight, finalTop, finalBottom;

    int left, right, i;
    int temp[ROW], sum, start, finish;

    // Set the left column
    for (left = 0; left < COL; ++left)
    {
        // Initialize all elements of temp as 0
        memset(temp, 0, sizeof(temp));

        // Set the right column for the left column set by outer loop
        for (right = left; right < COL; ++right)
        {
            // Calculate sum between current left and right for every row 'i'
            for (i = 0; i < ROW; ++i)
                temp[i] += M[i][right];

            // Find the maximum sum subarray in temp[]. The kadane() function
            // also sets values of start and finish.  So 'sum' is sum of
            // rectangle between (start, left) and (finish, right) which is t
            //  maximum sum with boundary columns strictly as left and right.
            sum = kadane(temp, &start, &finish, ROW);

            // Compare sum with maximum sum so far. If sum is more, then upda
            // maxSum and other output values
            if (sum > maxSum)
            {
                maxSum = sum;
                finalLeft = left;
                finalRight = right;
                finalTop = start;
                finalBottom = finish;
            }
        }
    }

    // Print final values
    printf("(Top, Left) (%d, %d)\n", finalTop, finalLeft);
    printf("(Bottom, Right) (%d, %d)\n", finalBottom, finalRight);
    printf("Max sum is: %d\n", maxSum);
}

// Driver program to test above functions
int main()
{
    int M[ROW][COL] = {{1, 2, -1, -4, -20},
                       {-8, -3, 4, 2, 1},
                       {3, 8, 10, 1, 3},
                       {-4, -1, 1, 7, -6}
                      };

    findMaxSum(M);

    return 0;
}
```

Output:

```
(Top, Left) (1, 1)
(Bottom, Right) (3, 3)
Max sum is: 29
```

Time Complexity: O(n^3)

This article is compiled by Aashish Barnwal. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Topics:

- Find Union and Intersection of two unsorted arrays
- Pythagorean Triplet in an array
- Maximum profit by buying and selling a share at most twice
- Design a data structure that supports insert, delete, search and getRandom in constant time
- Print missing elements that lie in range 0 – 99
- Iterative Merge Sort
- Group multiple occurrence of array elements ordered by first occurrence
- Given a sorted and rotated array, find if there is a pair with a given sum

Tags: Dynamic Programming

Tweet 3 g+1 6

Writing code in comment? Please use **ideone.com** and share the link here.

71 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

jhantu • 13 days ago

this link also has same algo but easy explanation.
http://tech-queries.blogspot.c...

Reply • Share ›

enunciate • 2 months ago

doesnot give rite indexes for

int M[ROW][COL] = { {INT_MIN, 1, 1, INT_MIN,INT_MIN},
{INT_MIN, 1, 1, 1, INT_MIN},
{INT_MIN, INT_MIN, 1, 1, INT_MIN},
{INT_MIN, 1, 1, 1, 1}
};

(Top, Left) (2, 1)
(Bottom, Right) (3, 4)
Max sum is: 6

Reply • Share ›

Aditya Goel • 3 months ago

Can somebody care to explain how this is a dynamic programming problem?

Reply • Share ›

vaibhav Aditya Goel • 2 months ago

Consider just the inner two loops
for when right =0
temp[0]=M[0][0]
temp[1]=M[0][1] and so on

when right =1
temp[0]= M[0][0]+M[0][1];
temp[1]=M[0][1]+M[1][1]

On each of these levels we run Kadane's algo to find max sum subarray
So you see M[0][1] gets added to M[0][0] and M[1][1] gets added to M[0][1] and we use this to get to the final result, So previous values to being stored and used to get to the final result which is precisely what we do in Dp

• Reply • Share ›

Aditya Goel vaibhav • 2 months ago

Can we represent this problem in terms of overlapping subproblems and optimal substructure?

• Reply • Share ›

Hardhik Mallipeddi • 4 months ago

Hey, @geeksforgeeks @aashishbarnwal I think in the kadane's algo part of the code:
" // There is at-least one non-negative number
if (*finish != -1)
return maxSum;
"

I believe finish will not be -1 if there is atleast one number > INT_MIN. since we check if sum>maxsum inside the for loop and maxsum's initial value is INT_MIN and sum's initial value is 0. So the finish index is update even for any negative element which is greater than INT_MIN in the array.

Please look into this and modify the code if needed.

• Reply • Share ›

rverma • 4 months ago

I am using kadane Algo with time complexity is O(n^2) and Auxillery Space is O(1).
Approach is:
1.run loop column wise and apply kadane algo in each column for finding a column which has maxContinousSubarray which is bigger among all column with max sum.
let start1 and end1 is subarray starting and ending points which we get from 1st traversing.start1 is 1st row and end1 is last row of final 2DsubArray which look like a strip.

2. again run loop column wise with running internal loop i=start1 to end1 and consider sum of all element of individual column is single element,apply kadane algo for finding maxSumContinous subarray with max sum.this max contain final sum of 2DSubarray and start2 and end2 of this SubArray are starting and ending column of final 2DsubArray.

top-leftmost point is (start1,start2)
bottom-rightmost point is (end1,end2)
and last max is sum of that 2DSubAray.
http://ideone.com/gZiw06

• Reply • Share ›

fdcp rverma • 4 months ago

Try this case:
{ {-1, -1, 3, 3},
{-1, -1, 3, 3},
{10, -1, -1, -1},
{-1, -1, -1, -1}};

Print 10, but the answer should be 12

• Reply • Share ›

rverma fdcp • 3 months ago

thanks for providing this test case...i updated the code by handling case i.e we have max sum but smaller length in colum and another case is sum is smaller but length of continous +ve num is bigger in colum.

http://ideone.com/KYbiTC

• Reply • Share ›

Kenneth • 4 months ago

My solution, where we pre-calculate the sum between different lines. Time complexity is O(n^3) and space complexity is O(n^2).

http://ideone.com/FyGGOq

• Reply • Share ›

knowledgefree • 7 months ago

memset has complexity!

• Reply • Share ›

Tyrion • 7 months ago

O(n^6) and O(n^4) implementation in java
http://ideone.com/gbjkLe

1 • Reply • Share ›

Guest • 8 months ago

Here is another method, i think it is easier one: Given array A[N][N],
1) Make a table Sum[N+1][N+1]
2) Initialize Sum[0][i] = A[0][i] for 0<=i<n 3)="" initialize="" sum[i][0]="A[i][0]" for="" 0<i<n="" 4)="" calculate="" sum[i][j]="A[i][j]" +="" max(sum[i-1][j-1],="" sum[i-1][j],="" sum[i][j-1])="" for="" 0<i,j<n="" 5)="" find="" position="" of="" maximum="" element="" in="" sum[][],="" let="" it="" be="" x="" and="" y.="" 6)="" x1="X," y1="Y" 7)="" while(x1="">=0 && Sum[X1][Y] >0) X1 = X1-1;
8) while(Y1>=0 && Sum[X][Y1] >0) Y1 = Y1-1;
9) Print all the elements of array A[][] from (X1,Y1) to (X,Y).
What is the problem with this algorithm? I am also finding the correct answer for this also. and It work in O(n^2) time.

4 • Reply • Share ›

NP-EASY • 9 months ago

for (i = 0; i < ROW; ++i)
temp[i] += M[i][right];

This loop is completely unnecessary

• Reply • Share ›

Rakesh Mondal NP-EASY • 8 months ago

U mad bro ?? -_-
each col element in new iteration is being added to the temp array so that kadane's can be calculated. -_- .

for example in this matrix in iteration 1 of outer loop

147
258
369

inner loop iterations :
1) temp = (1,2,6) and do kadanes
2) temp = (5, 7, 9) and do kadanes
3) temp = (12, 15, 18) and do kadanes

"for (i = 0; i < ROW; ++i)
temp[i] += M[i][right];"
adds current col elemnts to prev temp array

2 • Reply • Share ›

Navneet Verma NP-EASY • 8 months ago

How this loop is unnecessary???

• Reply • Share ›

Guest ↪ Navneet Verma • 8 months ago

I also think so , we can send the 1D array directly without making its copy in the temp array.

• Reply • Share ›

Deepanshu Agrawal ↪ Guest • 5 months ago

that is not just the 1 column which is being sent it's the sum of columns from left to right.

• Reply • Share ›

xinyuan • 10 months ago

isn't it O(N^4) still for kadane's algo? for findMaxsum theres already three for loops and the kadanes function has one for loop.

• Reply • Share ›

np ↪ xinyuan • 10 months ago

it is n^2*(n+n) : n^2 for outer for loop and (n+n) for inner and kadanes loop so total is: so O(n^3)

• Reply • Share ›

Whatever • a year ago

Time Complexity is precisely O(Col^2 * Row) !!

1 • Reply • Share ›

Guest • a year ago

Time Complexity is precisely O(Col^2 * Row) :)

• Reply • Share ›

Guest • a year ago

Hi, there is an O(n^2) algorithm for this using O(n^2) extra space.

Input :- given matrix a[][] of dimension m*n.

Algo:-

1) create a new matrix n[][] of dimension m*n.

2) n[i][j] = for k = 0 to i (sum of a[k][j]). That is the new matrix is the column wise sum matrix.

3) Now apply the well known standard "Max rectangle in histogram" algorithm on each row of the n[][].
http://www.geeksforgeeks.org/l...

a) Compare the curr_max with max.

return max.

Time complexity - O(n^2).

Reply • Share ›

jafar Guest • a year ago

I don't see how that will give us the desired result.since applying the max rectangle might use 'a fraction of a slot' e.g: using 3 out of a slot with value 4.or using different sets of rows for each column,thus not producing a rectangle.

Reply • Share ›

Aditya • a year ago

This will not work for matrix
{ {-1,-1,-1},
{9, 9, -1},
{-1,-1,-1}}.
It will give result as 17 but the answer should be 18.

Reply • Share ›

meh Aditya • a year ago

It works, I tried the code and it returns 18.

Reply • Share ›

Mythreya J L • a year ago

Can be reduced to O(n^2).

1. Find integral "image" representation of the matrix: O(n^2)
2. Find the position of the maximum element, (x1,y1) in the integral image: O(n^2)

3. Find the position of the minimum element, (x0,y0) that appears strictly before the maximum element found, which means it has to be at a lower row AND lower column: O(n^2).

Maximum subarray is (x0:x1, y0:y1), time complexity: O(n^2)+O(n^2)+O(n^2) = O(n^2)

Reply • Share ›

jerrym Mythreya J L • a year ago

Unfortunately, there is no guarantee that the maximum element in the integral image is also the last element of the maximum subarray.

2 • Reply • Share ›

Sumit Khanna • a year ago

Hey!...time complexity for BRUTE FORCE or NAIVE is O(n^6) and not O(n^4)....you fix all N^2 toplefts and ~ N^2 bottomRights for every topleft,,and N^2 for calculating sum of every considered rectangle of the N^4 possible rectangles...thus total time is O(n^6) considered rectangle of the N^4 possible rectangles,,,thus total time is O(n^6) ...

3 ^ | ⌄ • Reply • Share ›

Tyrion ➦ Sumit Khanna • 7 months ago

Look here for both O(n^6) and O(n^4) implementation. Plain bruteforce is O(n^6) but with DP to save cumulative sum of numbers we can do it in O(n^4)
http://ideone.com/gbjkLe

^ | ⌄ • Reply • Share ›

Bandi Sumanth ➦ Sumit Khanna • 8 months ago

It is not a pure brainless bruteforce. For a fixed top left and for a fixed top right -> the sum is calculated in constant time using the previous calculated values... think about it, its very easy

Its just like for a 1D array, the bruteforce solution is O(n^2) but not O(n^3)

^ | ⌄ • Reply • Share ›

AlienOnEarth ➦ Sumit Khanna • a year ago

Yes Sumit Khanna. Even I was thinking the same. I think you are right. It should be updated to o(n^6). Also confirmed on other websites.

@GeeksforGeeks:
Please consider this once and update if looks correct to you.

1 ^ | ⌄ • Reply • Share ›

Bandi Sumanth ➦ AlienOnEarth • 8 months ago

It is not a pure brainless bruteforce. For a fixed top left and for a fixed top right -> the sum is calculated in constant time using the previous calculated values... think about it, its very easy.
So it is O(n^4) only

Its just like for a 1D array, the bruteforce solution is O(n^2) but not O(n^3).

^ | ⌄ • Reply • Share ›

helper ➦ Bandi Sumanth • 7 months ago

hey bandi, these people are not getting what you are saying. let me put it this way.

step 1: given a matrix A you do some precomputing:
find the cumulative matrix C[m*n] where C[i,j]=sum of all a[l,m]; i<=l, j<=m. this requires just O(mn) or O(n^2) by formula: c[i,j]=a[i][j]+c[i][j-1]+c[i-1][j]-c[i-1][j-1] except first row and first column, which can be done considering the sum till that point in an array.

step 2: now if you ask me sum of any rectangle, i can tell you that in just O(1) i.e. constant time... using inclusion exclusion principle (i am giving

O(1) i.e. constant time... using inclusion exclusion principle.(I am giving you the formula, please take a sample A and C to verify):
sum of sub-matrix of A starting at x,y and ending at z,w: c[z,w]+c[x,y]-c[x,w]-c[z,y]

Remark: Formula does the task in constant time.

• Reply • Share ›

intdydx • 2 years ago
I implemented an FFT-based solution here:
https://github.com/thearn/maxi...

• Reply • Share ›

Sumant Kumar Dev • 2 years ago
Neha Modi WA for {{1,-9,-5,2,8},

{11, 4, 3, 7, 4},

{1, -5, -7, 8, -3},

{2, 3, -4, 5, -2}.

};.

2 • Reply • Share ›

Neha Modi • 2 years ago
of order n^2.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
int max(int a, int b, int c).
{ if(a>b && a>c).
return a;.
if(b>a && b>c).
return b;.
return c;.
}
void main()
{ int arr[4][5]={{1,2,-1,-4,-20},{-8,-3,4,2,1},{3,8,10,1,3},{-4,-1,1,7,-6}}, s[4][5], i,j;.
int max1=0;.
for(i=0;i<4;i++).
s[i][0]=arr[i][0];.
for(j=0;j<5;j++).
s[0][j]=arr[0][j];.
```

see more

• Reply • Share ›

Yash Pareek • 2 years ago

@pankaj....it is like if u have array from left to right as:
1 2
3 4
5 6

then temp={(1+2),(3+4),(5+6)};

1 • Reply • Share ›

Pankaj Goyal • 2 years ago

can anybody tell me how are the values filled in the temp[] array? horizontally or vertically?

• Reply • Share ›

fenglvming • 2 years ago

```
 /* Paste your code here (You may delete these lines if not writing code) */
for (left = 0; left < COL; ++left)
    {
        // Initialize all elements of temp as 0
        memset(temp, 0, sizeof(temp));

        // Set the right column for the left column set by outer loop
        for (right = left; right < COL; ++right)
        {
            // Calculate sum between current left and right for every row 'i'
            for (i = 0; i < ROW; ++i)
                temp[i] += M[i][right];

            // Find the maximum sum subarray in temp[]. The kadane() function
            // also sets values of start and finish.  So 'sum' is sum of
            // rectangle between (start, left) and (finish, right) which is the
            //  maximum sum with boundary columns strictly as left and right.
            sum = kadane(temp, &start, &finish, ROW);
```

see more

• Reply • Share ›

Born Actor • 2 years ago

```
 #include <iostream>
#include<string>
#include<sstream>
#include<iomanip>
# include <stdio.h>
# include <math.h>
#include <vector>
#include <stdlib.h>
using namespace std;
int n_r;
int n_c;
int n;
int a[50];
pair < int , pair < int , int > > function(int i, int j);
 pair < int , pair < pair < int,  int > , pair < int ,int > > > answer;
pair < int , pair < int , int > > function2(int i, int m, int j);
pair < int , pair < int , int > > max(pair < int , pair < int , int >  > a,pair < int , pai
int b[50][50];
```

see more

• Reply • Share ›

GeeksforGeeks • 2 years ago

Ashish Anand: Thanks for suggesting the fix. We have updated the post.

• Reply • Share ›

logic_bomber • 2 years ago

Why is it under 'Dynamic Programming'? Do help me understand what property of Dynamic Programming comes here.

• Reply • Share ›

GeeksforGeeks • 2 years ago

Thanks for pointing this out. We will look into this and update the post.

• Reply • Share ›

Chaitanya T V Krishna GeeksforGeeks • a year ago

We do not escape n2 complexity in column. But we use Kandane's algorithm during computing maximum sum contiguous rows to convert n2 to n.

• Reply • Share ›

Vikas Singla • 2 years ago

GeeksforGeeks.....above code gives wrong sol for my following comments....
correct if I&#039m not wrong.

• Reply • Share ›

Vikas Singla • 2 years ago

#define ROW 6.
#define COL 1.
int main()

{

int M[ROW][COL] = {-1,-2,4,-3,-2, 2.

};.

findMaxSum(M);.

return 0;.
}

solution with this is.
(Top, Left) (5, 0).
(Bottom, Right) (2, 0).
Max sum is: 4.

which is wrong I think so....

• Reply • Share ›

Paparao Veeragandham • 2 years ago

```
 Given a matrix of both +ve and -ve numbers, find out the maximum sum sub matrix. First of


int s[ROW][COL];
 void computeSumMatrix(int a[][COL], int r, int c) {
  for (int i = 0; i < r; i++)
   if (i == 0)
    s[i][0] = a[i][0];
   else
    s[i][0] = s[i - 1][0] + a[i][0];

  for (int j = 0; j < c; j++)
   if (j == 0)
    s[0][j] = a[0][j];
   else
    s[0][j] = s[0][j - 1] + a[0][j];

  for (int i = 1; i < r; i++) {
   for (int j = 1; j < c; j++) {
```

see more

1 • Reply • Share ›

rkl • 2 years ago
The lines

```
 // Calculate sum between current left and right for every row 'i'
```

```
        for (i = 0; i < ROW; ++i)
            temp[i] += M[i][right];
```

is wrong.

We should calculate temp[i] = M[i][left] + ... + M[i][right]
which should be done in a loop.

```
        for (i = 0; i < ROW; ++i)
           for(int j = left; j <=right; j++)
               temp[i] += M[i][j];
```

• Reply • Share ›

rkl rkl • 2 years ago

I think I got the point. Please ignore my previous comment.
In the first iteration when right=left, there is only 1 column in temp per i.
temp[i] = 0 (temp is initialized to 0) + M[i][right or left, its same in first iteration]

In 2nd iteration, when right=left+1,
temp[i] = old_temp[i] (which is M[i][left]) + M[i][right]

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Follow @GeeksforGeeks

- ## Recent Comments

  - lt_k

    i need help for coding this function in java...

    Java Programming Language · 1 hour ago
  - Piyush

    What is the purpose of else if (recStack[*i])...

    Detect Cycle in a Directed Graph · 1 hour ago
  - Andy Toh

    My compile-time solution, which agrees with the...

    Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago
  - lucy

    because we first fill zero in first col and...

    Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago
  - lucy

    @GeeksforGeeks i don't n know what is this long...

    Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago
  - manish

    Because TAN is not a subsequence of RANT. ANT...

    Given two strings, find if first string is a subsequence of second · 3 hours ago
-