GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

How to print maximum number of A's using given four keys

This is a famous interview question asked in Google, Paytm and many other company interviews.

Below is the problem statement.

```
Imagine you have a special keyboard with the following keys:
Key 1:  Prints 'A' on screen
Key 2: (Ctrl-A): Select screen
Key 3: (Ctrl-C): Copy selection to buffer
Key 4: (Ctrl-V): Print buffer on screen appending it
                 after what has already been printed.

If you can only press the keyboard for N times (with the above four
keys), write a program to produce maximum numbers of A's. That is to
say, the input parameter is N (No. of keys that you can press), the
```

```
output is M (No. of As that you can produce).
```

Examples:

```
Input:  N = 3
Output: 3
We can at most get 3 A's on screen by pressing
following key sequence.
A, A, A

Input:  N = 7
Output: 9
We can at most get 9 A's on screen by pressing
following key sequence.
A, A, A, Ctrl A, Ctrl C, Ctrl V, Ctrl V

Input:  N = 11
Output: 27
We can at most get 27 A's on screen by pressing
following key sequence.
A, A, A, Ctrl A, Ctrl C, Ctrl V, Ctrl V, Ctrl A,
Ctrl C, Ctrl V, Ctrl V
```

We strongly recommend to minimize your browser and try this yourself first.

Below are few important points to note.
a) For N < 7, the output is N itself.
b) Ctrl V can be used multiple times to print current buffer (See last two examples above).

The idea is to compute the optimal string length for N keystrokes by using a simple insight. The sequence of N keystrokes which produces an optimal string length will end with a suffix of Ctrl-A, a Ctrl-C, followed by only Ctrl-V's (For N > 6).
The task is to find out the break=point after which we get the above suffix of keystrokes. Definition of a breakpoint is that instance after which we need to only press Ctrl-A, Ctrl-C once and the only Ctrl-V's afterwards to generate the optimal length. If we loop from N-3 to 1 and choose each of these values for the break-point, and compute that optimal string they would produce. Once the loop ends, we will have the maximum of the optimal lengths for various breakpoints, thereby giving us the optimal length for N keystrokes.

Below is C implementation based on above idea.

```
/* A recursive C program to print maximum number of A's using
   following four keys */
#include<stdio.h>

// A recursive function that returns the optimal length string
// for N keystrokes
int findoptimal(int N)
{
    // The optimal string length is N when N is smaller than 7
    if (N <= 6)
        return N;

    // Initialize result
    int max = 0;
```

```
    // TRY ALL POSSIBLE BREAK-POINTS
    // For any keystroke N, we need to loop from N-3 keystrokes
    // back to 1 keystroke to find a breakpoint 'b' after which we
    // will have Ctrl-A, Ctrl-C and then only Ctrl-V all the way.
    int b;
    for (b=N-3; b>=1; b--)
    {
            // If the breakpoint is s at b'th keystroke then
            // the optimal string would have length
            // (n-b-1)*screen[b-1];
            int curr = (N-b-1)*findoptimal(b);
            if (curr > max)
                max = curr;
     }
     return max;
}

// Driver program
int main()
{
    int N;

    // for the rest of the array we will rely on the previous
    // entries to compute new ones
    for (N=1; N<=20; N++)
        printf("Maximum Number of A's with %d keystrokes is %d\n",
               N, findoptimal(N));
}
```

Output:

```
Maximum Number of A's with 1 keystrokes is 1
Maximum Number of A's with 2 keystrokes is 2
Maximum Number of A's with 3 keystrokes is 3
Maximum Number of A's with 4 keystrokes is 4
Maximum Number of A's with 5 keystrokes is 5
Maximum Number of A's with 6 keystrokes is 6
Maximum Number of A's with 7 keystrokes is 9
Maximum Number of A's with 8 keystrokes is 12
Maximum Number of A's with 9 keystrokes is 16
Maximum Number of A's with 10 keystrokes is 20
Maximum Number of A's with 11 keystrokes is 27
Maximum Number of A's with 12 keystrokes is 36
Maximum Number of A's with 13 keystrokes is 48
Maximum Number of A's with 14 keystrokes is 64
Maximum Number of A's with 15 keystrokes is 81
Maximum Number of A's with 16 keystrokes is 108
Maximum Number of A's with 17 keystrokes is 144
Maximum Number of A's with 18 keystrokes is 192
Maximum Number of A's with 19 keystrokes is 256
Maximum Number of A's with 20 keystrokes is 324
```

The above function computes the same subproblems again and again. Recomputations of same subproblems can be avoided by storing the solutions to subproblems and solving problems in bottom up manner.

Below is Dynamic Programming based C implementation where an auxiliary array screen[N] is used to store result of subproblems.

```
/* A Dynamic Programming based C program to find maximum number of A's
   that can be printed using four keys */
#include<stdio.h>

// this function returns the optimal length string for N keystrokes
int findoptimal(int N)
{
    // The optimal string length is N when N is smaller than 7
    if (N <= 6)
        return N;

    // An array to store result of subproblems
    int screen[N];

    int b;  // To pick a breakpoint

    // Initializing the optimal lengths array for uptil 6 input
    // strokes.
    int n;
    for (n=1; n<=6; n++)
        screen[n-1] = n;

    // Solve all subproblems in bottom manner
    for (n=7; n<=N; n++)
    {
        // Initialize length of optimal string for n keystrokes
        screen[n-1] = 0;

        // For any keystroke n, we need to loop from n-3 keystrokes
        // back to 1 keystroke to find a breakpoint 'b' after which we
        // will have ctrl-a, ctrl-c and then only ctrl-v all the way.
        for (b=n-3; b>=1; b--)
        {
            // if the breakpoint is at b'th keystroke then
            // the optimal string would have length
            // (n-b-1)*screen[b-1];
            int curr = (n-b-1)*screen[b-1];
            if (curr > screen[n-1])
                screen[n-1] = curr;
        }
    }

    return screen[N-1];
}

// Driver program
int main()
{
    int N;
```

```
    // for the rest of the array we will rely on the previous
    // entries to compute new ones
    for (N=1; N<=20; N++)
        printf("Maximum Number of A's with %d keystrokes is %d\n",
               N, findoptimal(N));
}
```

Output:

```
Maximum Number of A's with 1 keystrokes is 1
Maximum Number of A's with 2 keystrokes is 2
Maximum Number of A's with 3 keystrokes is 3
Maximum Number of A's with 4 keystrokes is 4
Maximum Number of A's with 5 keystrokes is 5
Maximum Number of A's with 6 keystrokes is 6
Maximum Number of A's with 7 keystrokes is 9
Maximum Number of A's with 8 keystrokes is 12
Maximum Number of A's with 9 keystrokes is 16
Maximum Number of A's with 10 keystrokes is 20
Maximum Number of A's with 11 keystrokes is 27
Maximum Number of A's with 12 keystrokes is 36
Maximum Number of A's with 13 keystrokes is 48
Maximum Number of A's with 14 keystrokes is 64
Maximum Number of A's with 15 keystrokes is 81
Maximum Number of A's with 16 keystrokes is 108
Maximum Number of A's with 17 keystrokes is 144
Maximum Number of A's with 18 keystrokes is 192
Maximum Number of A's with 19 keystrokes is 256
Maximum Number of A's with 20 keystrokes is 324
```

Thanks to **Gaurav Saxena** for providing the above approach to solve this problem.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- Write an iterative O(Log y) function for pow(x, y)
- Visa Interview Experience | Set 6 (On-Campus)

Tags: Dynamic Programming

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

Recommend Share Sort by Newest

Join the discussion…

ganeshprabhu1994 • 15 days ago
http://ideone.com/EcbqvD
• Reply • Share ›

Guest ganeshprabhu1994 • 2 days ago
Can you explain your logic?
• Reply • Share ›

GOPI GOPINATH ganeshprabhu1994 • 10 days ago
explain ur logic :)
• Reply • Share ›

This comment was deleted.

Venkat Raman Guest • 19 days ago
O
• Reply • Share ›

Abhijeet Sachdev • 25 days ago
There is no need of running the loop from "N-3 to 1". It is sufficient to run from "N-3 to N-5"
• Reply • Share ›

goneCrazy Abhijeet Sachdev • 25 days ago
I think it should be N-3 to N-6
• Reply • Share ›

Piotr goneCrazy • 23 days ago
N-5 is sufficient. (the loop goes through N-3, N-4 and N-5 inclusive)

BTW this comment:

// For any keystroke n, we need to loop from n-3 keystrokes
// back to 1 keystroke to find a breakpoint 'b' after which we
// will have ctrl-a, ctrl-c and then only ctrl-v all the way.

is a bit wrong. The only sensible combinations to do is:
ctrl+a ctrl+c ctrl+v ctrl+v (N-5)
ctrl+a ctrl+c ctrl+v (N-4)

ctrl+v ctrl+v (N-3)

there is no way that a longer sequence of ctrl+v gives anything better.

• Reply • Share ›

Guest • 25 days ago

There is no need of running the lob from "N-3 to 1". It is sufficient to run from "N-3 to N-5"

• Reply • Share ›

Guest • a month ago

```
private int findMax(int n) {
// TODO Auto-generated method stub
int dp[] = new int[n+1];
for(int i=1;i<7;i++){
dp[i] = i;
}
if(n<7){
return n;
}
int cv = 3;
for(int i =7;i<=n;i++){
dp[i] = Math.max(3*dp[i-4], dp[i-1]+(dp[i-1] - dp[i-2]));
/*/*/
}
return dp[n];
}
```

• Reply • Share ›

Piotr ➚ Guest • 23 days ago

Will fail for 9 giving 15 instead of 16. Probably missed 2*dp[i-3]

• Reply • Share ›

Shankar Kshetry • a month ago

this can be solved with minimum complexity as below:

```
int f(int n)

{

if (n<= 6)
return n;
else return Math.max(f((n-1)-3)*3, f((n-1)-4)*4) ;

}
```

• Reply • Share ›

Piotr ➚ Shankar Kshetry • 23 days ago

In terms of lines needed to code it may be less complex, but it still does many reduntant computations and is WAY slower than any DP shown here. But it can be converted to DP very easily just using standard array

▲ | ▼ • Reply • Share ›

Guest • a month ago

```
def maxA(n):
max = 0
if n < 7:
return n
max,k = 3,3
for i in range(3,n):
if k >= n:
break;
if n-k >= 3:
k=k+3
copymax=max;
max=max+copymax
else:
max = max+copymax
k=k+1
return max
```

▲ | ▼ • Reply • Share ›

kaushik Lele • 2 months ago

I understood the logic but could not quickly understand the line
(N-b-1)*findoptimal(b);
Especially why (N-b-1) ?

Then I realized the logic as follows :-
If you are at breakpoint - b; then till that point you have achieved findoptimal(b) number of A's
Now you have (N-b) chances left. Now you will do ctrl+a and ctrl+c.
So you have findoptimal(b) in your buffer which can be used for copy.
After these two key press you have N-b-2 chances left.
So number of times you can do ctrl+v is N-b-2
So you will get further (N-b-2)*buffer i.e. (N-b-2)*findoptimal(b) of A's

totoal number of A's =
A's till break point b + A's after breakpoint b
=findoptimal(b)+ (N-b-2)*findoptimal(b)
=(N-b-2+1)findoptimal(b)
=(N-b-1)findoptimal(b)

I hope it help other readers

• Reply • Share ›

Venkat Raman ➚ kaushik Lele • 16 days ago

@GeeksforGeeks kindly do add this explaination along with the code.. It'll help other users

• Reply • Share ›

Venkat Raman ➚ kaushik Lele • 16 days ago

Thanks man for such a nice explaination.. Could you explain about the TIME and SPACE COMPLEXITY of this program

• Reply • Share ›

Abhijeet Sachdev ➚ kaushik Lele • 25 days ago

Thanks for such a nice explanation

• Reply • Share ›

apurva ➚ kaushik Lele • a month ago

awesome explanation!!!

• Reply • Share ›

Jason ➚ kaushik Lele • a month ago

THIS IS SO HELPFUL!!

• Reply • Share ›

kaushik Lele ➚ kaushik Lele • 2 months ago

@GeeksforGeeks can you add this explanation to main article. This will help other readers.

• Reply • Share ›

Ankur Dwivedi ➚ kaushik Lele • a month ago

could u plz xpalin why... n-3 is used as starting point

• Reply • Share ›

Sujan Dey ➚ Ankur Dwivedi • a month ago

because if u want to use ctrl a, ctrl c nd ctrl v , u have to use three keystrokes.....

• Reply • Share ›

Guest • 2 months ago

Can you please elaborate on below logic:

// if the breakpoint is at b'th keystroke then

```
// the optimal string would have length

// (n-b-1)*screen[b-1];

int curr = (n-b-1)*screen[b-1]
```

• Reply • Share ›

kaushik Lele Guest • 2 months ago

Please check my above comment

• Reply • Share ›

Guest kaushik Lele • 2 months ago

Thanks

• Reply • Share ›

Nitesh • 2 months ago

The problem seems to follow a pattern. 27*4 = 108(max of 16), 36*4 = 144(max of 17), 48*4 = 192(max of 18), 64*4 = 256 (max of 19), 81*4 = 324(max of 20) and so on...

• Reply • Share ›

prashant jha • 2 months ago

here is the recursive solution

```
#include <iostream>
using namespace std;
int maxim(int a,int b)
{
return a>b?a:b;
}
int maxim(int a,int b,int c)
{
return maxim(maxim(a,b),c);
}
int prin_max_a(int n,int cnt,int buff)
{
if(n<=0)
return 0;
return maxim(1+prin_max_a(n-1,cnt+1,buff),
prin_max_a(n-2,cnt,cnt),
buff+prin_max_a(n-1,cnt+buff,buff));
}
int main(int argc, char const *argv[])
{
int n;
```

```
int n;
cout<<"enter the value of n .\n";
cin>>n;
int cnt=0;int buff=0;
cout<<prin_max_a(n,cnt,buff); return="" 0;="" }="">
```

• Reply • Share ›

SANDIPAN • 2 months ago

N = 7 should not be 9. Its actually 7.
Since a,a,a,a,Ctrl+A,Ctrl+C,Ctrl+V will result 4 a's rather than doubling it. Since the Ctrl+A selects all 4 a's and Ctrl+V will replace the previous 4 a's.

• Reply • Share ›

Cherish ➦ SANDIPAN • 2 months ago

It should be 9. Firstly, get AAA, then press ctrl-v for twice.

• Reply • Share ›

Sandipan • 2 months ago

One main thing is that I can be solve in O(n)
Since we are optimizing the sub-problems each time, we don't need to loop through n-3 to 1.

If we are looking it carefully, our last activity should be either two Ctrl+V or three Ctrl + V in order to get the maximum output. Instead of iterating through all items just iterate through n-3 to n-5 - in-order to get maximum output.

1 • Reply • Share ›

Manish ➦ Sandipan • 2 months ago

No, it can't be o(n).
That loop is actually doing, select copy and paste on all the rest screen[b], where b is less than n-3.
If you remove that loop, you will end up doing paste only operation which will result in wrong answer.

1 • Reply • Share ›

Cherish ➦ Sandipan • 2 months ago

Could you show your O(N) code? I can only think of an O(n^2) solution with DP.
Thanks!

1 • Reply • Share ›

Manish ➦ Cherish • a month ago

```
#include "bits/stdc++.h"

using namespace std;

int getMaxAs(int moves)
```

```
{
int* dp = new int [moves+1];
dp[0] = 0;
dp[1] = 1;
dp[2] = 2;

int inCopy = 0;
for(int i = 2; i <= moves; i++)
{
int num1 = dp[i-1] +1;
int num2 = 2*dp[i-3];
int num3 = dp[i-1] + inCopy;
int maxA = (num1 >= num2 ? (num1 >= num3 ? num1 : num3) : (num2 >= num3
2 num2 : num3));
```

see more

1 • Reply • Share ›

Guest • 2 months ago

```
import java.util.Scanner;

public class KeyStokeA {

public static int noOfStrings(int n)

{

if(n<7)

return n;

int no = 0;

for(int j=n/2;j<n-3;j++) {="" int="" f="(n-j-1)*noOfStrings(j);" if(f="">no)

no = f;

}

return no;
```

see more

• Reply • Share ›

Guest • 2 months ago

```
import java.util.Scanner;

/*
```

```
* To change this template, choose Tools | Templates

* and open the template in the editor.

*/

/**

*

* @author Cool Kid

*/

public class KeyStokeA {

/**
```

see more

• Reply • Share ›

sai • 2 months ago

why we need to loop n-3 to 1

• Reply • Share ›

Some many sai • 2 months ago

Because (n-3) + 3 = n and that we need at least 3 keystrokes to do something : we need to copy (takes 2 keystrokes) and paste at least once (takes 1 keystroke).

• Reply • Share ›

Shashi Jey • 2 months ago

can anyone print the sequence for 14,i am unable to understand the solution

• Reply • Share ›

Ram Shashi Jey • 2 months ago

the sequence is:

A A A A Ctrl A, Ctrl C, Ctrl V, Ctrl V, Ctrl V, Ctrl A, Ctrl C, Ctrl V, Ctrl V, Ctrl V

In simpler terms, for 9 key presses, you know that the best outcome is 16As. So..for 14, we still have 5 more key presses left...

Ctrl A, Ctrl C, Ctrl V, Ctrl V, Ctrl V ---these 5 produce best for 14.

• Reply • Share ›

Shashi Jey Ram • 2 months ago

how to guess how many a's will be for any number

• Reply • Share ›

Someone ➚ Shashi Jey • 2 months ago

There is a pattern, you can do without dynamic programming.

• Reply • Share ›

Shashi Jey ➚ Someone • 2 months ago

wat is that??

• Reply • Share ›

Someone ➚ Shashi Jey • 2 months ago

To compute the maximum number of As we can print, we have the following possibilities with N keystrokes :
* ending the keystrokes with Ctrl+A, Ctrl+C, Ctr+V
* ending the keystrokes with Ctrl+A, Ctrl+C, Ctrl+V, Ctrl+V
* etc.
* or pressing N times the letter A.

So, we consider each possibility to see what is better. If we end with Ctrl+A, Ctrl+C, Ctrl+V, etc. (M keystrokes), then the maximum number of As is produced when the preceding keystrokes (N-M) produced the maximum number of As. So, we have a recursive function here.

That's what is done in the first code. But doing that, we will recompute many times the maximum number of As for a given number of keystrokes. So, we can keep these numbers in memory instead. That's what is done in the second code.

• Reply • Share ›

Shashi Jey ➚ Someone • 2 months ago

can u take 14 as example and do every step according to code ,i find it very difficult to understand

https://www.facebook.com/shash...

u can msg me ur paperwork on my facebook id

plzz help me

• Reply • Share ›

Someone ➚ Shashi Jey • 2 months ago

I do not read code, sorry (I find it very ugly and having little interest).

• Reply • Share ›

Someone ➚ Someone • 2 months ago

And I don't have the time.

• Reply • Share ›

Someone ➚ Shashi Jey • 2 months ago

Dynamic programming? It's when you keep in memory the solutions of the problems you already have solved. It's what is done in the article.

You can look at my posts below, I explain the pattern (in the second reply of my post). Tell me if it's clear enough.

• Reply • Share ›

Ram ➚ Shashi Jey • 2 months ago

You have solve with dynamic programming. It checks for all possible combinations while storing the results for repeating sub problems.

• Reply • Share ›

PSK • 2 months ago

Didnt understand this part

```
// if the breakpoint is at b'th keystroke then
// the optimal string would have length
// (n-b-1)*screen[b-1];
int curr = (n-b-1)*screen[b-1];
```

• 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 · 2 hours ago

- Piyush

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

  Detect Cycle in a Directed Graph · 2 hours ago

- Andy Toh

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

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 2 hours 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