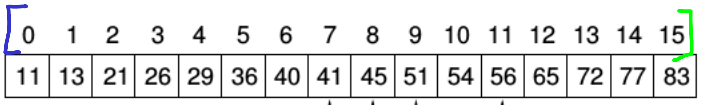
Binary Search

20170411

used some pictures and codes from
http://people.cs.vt.edu/shaffer/Book/C++3elatest.pdf
Data Structures and Algorithm Analysis
by Clifford A. Schaffer

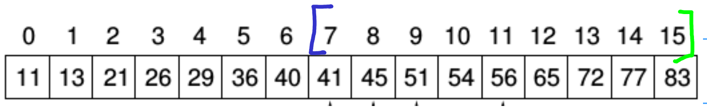


[0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15]
11	13	21	26	29	36	40	41	45	51	54	56	65	72	77	83

$\ell = 0$ $r = 15$ $i = \frac{1}{2}(\ell + r) = \frac{1}{2}(0 + 15) = 7$

$A[7] = 41 < 45$

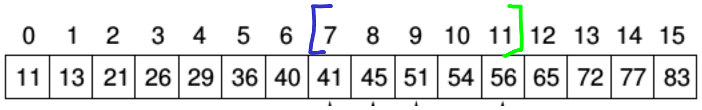
22



0	1	2	3	4	5	6	[7	8	9	10	11	12	13	14	15]
11	13	21	26	29	36	40	41	45	51	54	56	65	72	77	83

$\ell = 7$ $r = 15$ $i = \frac{1}{2}(\ell + r) = \frac{1}{2}(7 + 15) = 11$

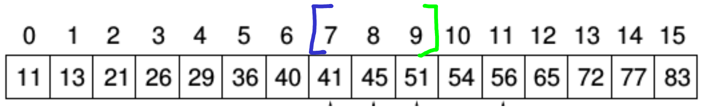
$45 < A[11] = 56$



0	1	2	3	4	5	6	[7	8	9	10	11]	12	13	14	15
11	13	21	26	29	36	40	41	45	51	54	56	65	72	77	83

$\ell = 7$ $r = 11$ $i = \frac{1}{2}(\ell + r) = \frac{1}{2}(7 + 11) = 9$

$45 < A[9] = 51$

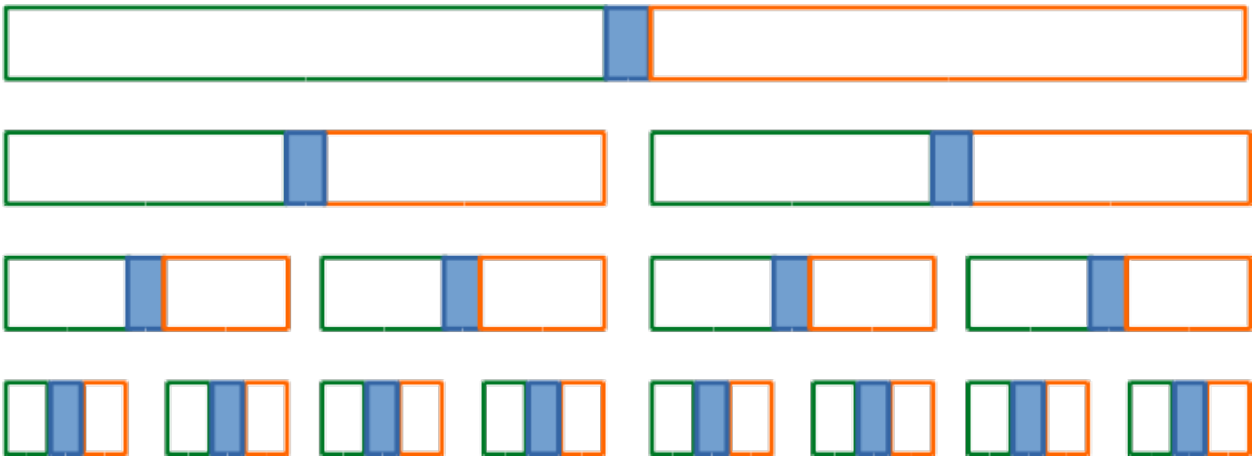


0	1	2	3	4	5	6	[7	8	9]	10	11	12	13	14	15
11	13	21	26	29	36	40	41	45	51	54	56	65	72	77	83

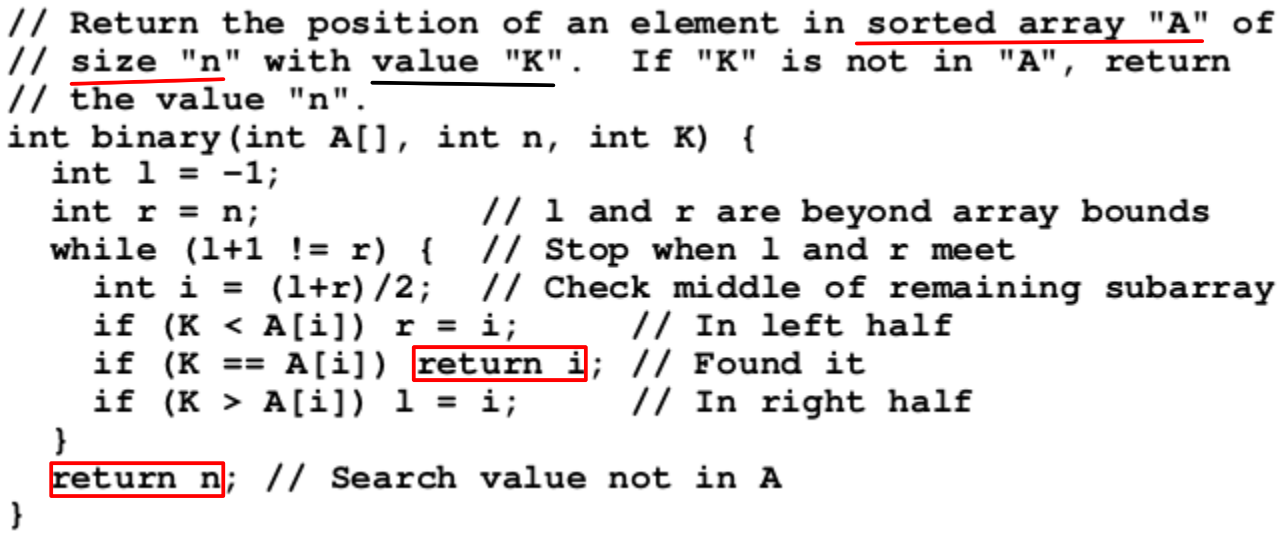
$\ell = 7$ $r = 9$ $i = \frac{1}{2}(\ell + r) = \frac{1}{2}(7 + 9) = 8$

$A[8] = 45$

$\boxed{\text{Index} = 8}$!



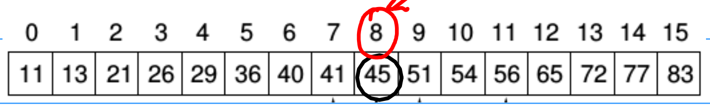
Search the location of the value K (index of A[ ])



```
// Return the position of an element in sorted array "A" of
// size "n" with value "K".  If "K" is not in "A", return
// the value "n".
int binary(int A[], int n, int K) {
  int l = -1;
  int r = n;            // l and r are beyond array bounds
  while (l+1 != r) {    // Stop when l and r meet
    int i = (l+r)/2;    // Check middle of remaining subarray
    if (K < A[i]) r = i;       // In left half
    if (K == A[i]) return i;   // Found it
    if (K > A[i]) l = i;       // In right half
  }
  return n; // Search value not in A
}
```

Let K=45

want to find

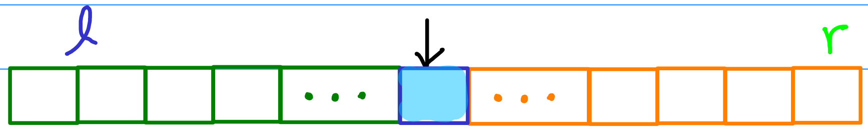


i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A[ ]	11	13	21	26	29	36	40	41	45	51	54	56	65	72	77	83

Sorted

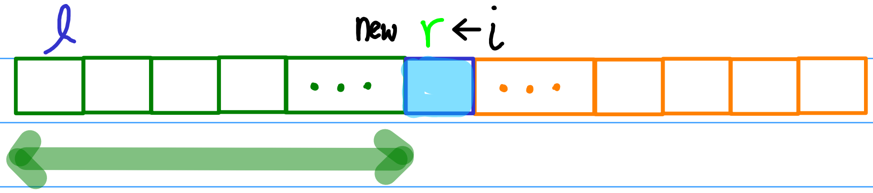
l : left r : right

$i = \frac{1}{2}(l + r)$



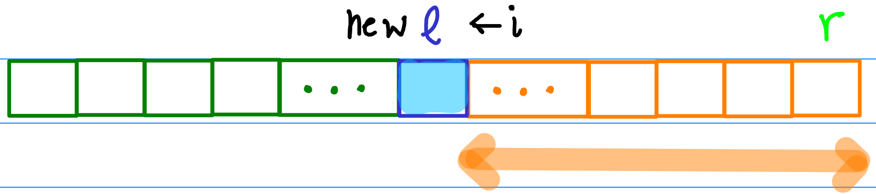
$< \; < \; < \; < \; A[i] \; < \; < \; < \; < \; <$ sorted already

case ① $\boxed{K < A[i]}$



next time search only this range

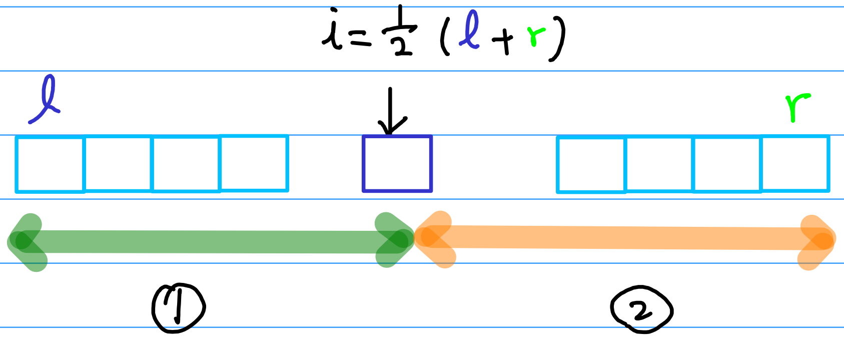
case ② $\boxed{A[i] < K}$



next time search only this range

case ③ $A[i] = K$

found the answer i



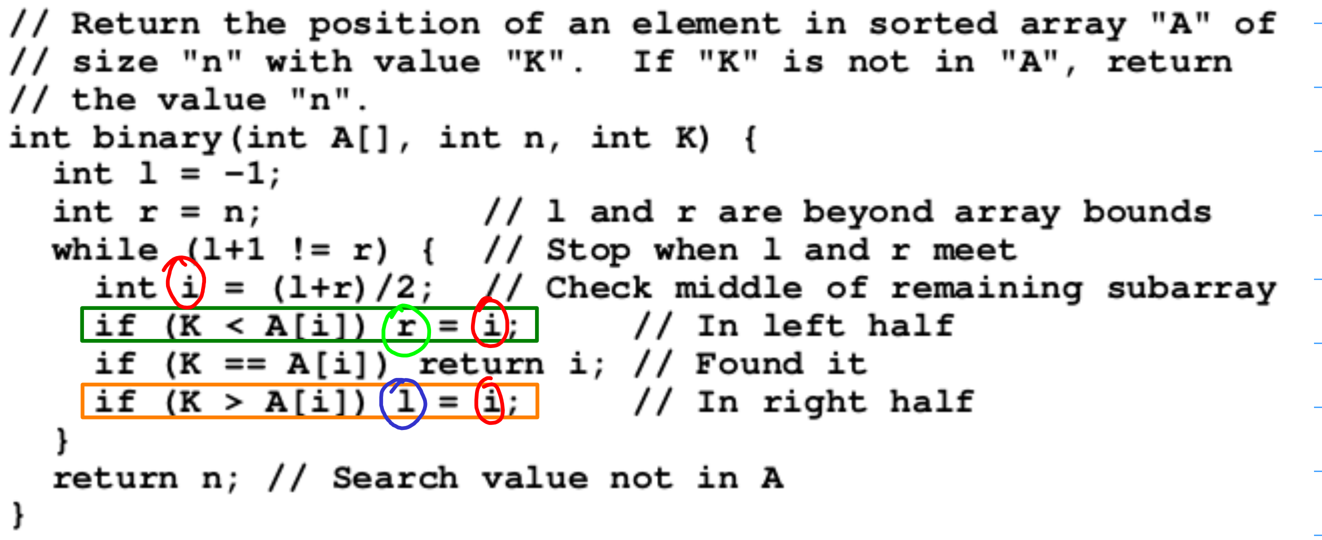
$K < A[i]$ the next search will be done over ①

$K > A[i]$ ②

because A[ ] is in the increasing order!

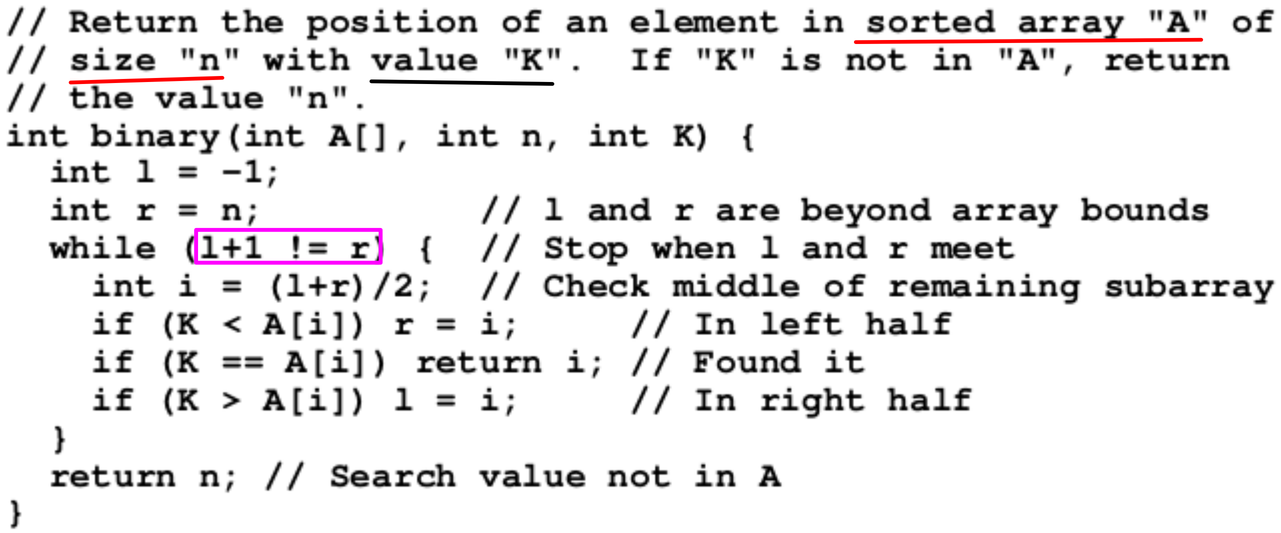
$K < A[i]$ the adjusted range $[l, i]$

$K > A[i]$ $[i, r]$



```
// Return the position of an element in sorted array "A" of
// size "n" with value "K".  If "K" is not in "A", return
// the value "n".
int binary(int A[], int n, int K) {
  int l = -1;
  int r = n;              // l and r are beyond array bounds
  while (l+1 != r) {      // Stop when l and r meet
    int i = (l+r)/2;      // Check middle of remaining subarray
    if (K < A[i]) r = i;        // In left half
    if (K == A[i]) return i;  // Found it
    if (K > A[i]) l = i;        // In right half
  }
  return n; // Search value not in A
}
```

Termination Condition



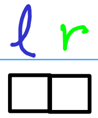
```
// Return the position of an element in sorted array "A" of
// size "n" with value "K".  If "K" is not in "A", return
// the value "n".
int binary(int A[], int n, int K) {
  int l = -1;
  int r = n;              // l and r are beyond array bounds
  while (l+1 != r) {      // Stop when l and r meet
    int i = (l+r)/2;      // Check middle of remaining subarray
    if (K < A[i]) r = i;       // In left half
    if (K == A[i]) return i;   // Found it
    if (K > A[i]) l = i;       // In right half
  }
  return n; // Search value not in A
}
```

l : left   <=   r : right

as the iteration goes on

l → moves to the right   ← r moves to the left

l += ●   r -= ●



last iteration

$l+1 = r$

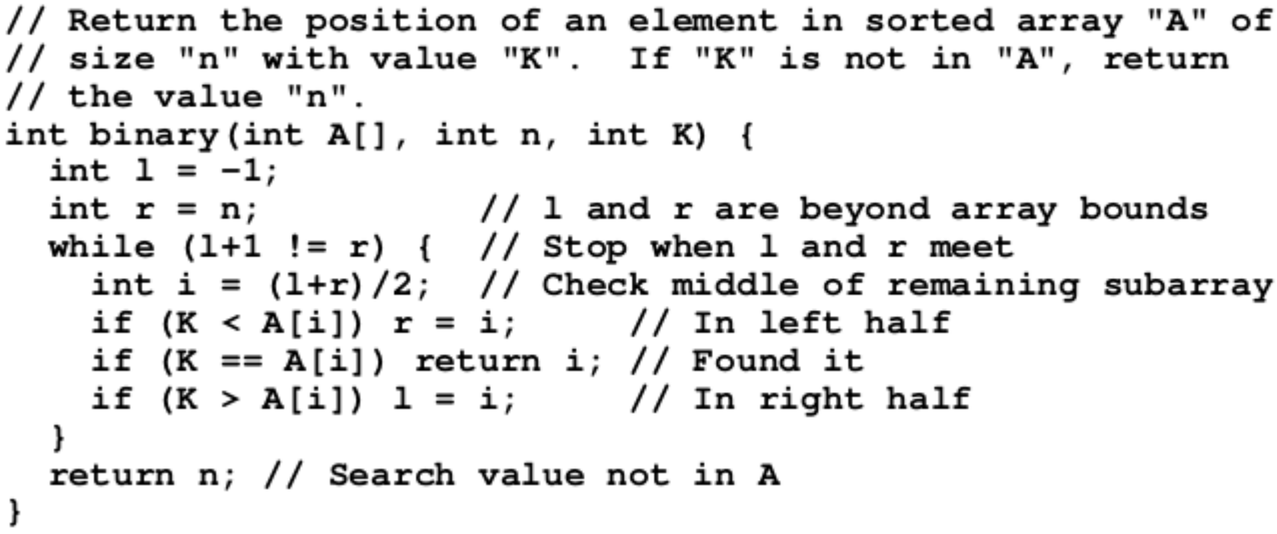
$\boxed{l <= r}$

initial condition.

$l = 0$
$r = n-1$ } n element array.

Since the index $\textcircled{i}$ is used
don't have to use

new $r \leftarrow i-1$
ne $l \leftarrow i+1$



```
// Return the position of an element in sorted array "A" of
// size "n" with value "K".  If "K" is not in "A", return
// the value "n".
int binary(int A[], int n, int K) {
  int l = -1;
  int r = n;              // l and r are beyond array bounds
  while (l+1 != r) {      // Stop when l and r meet
    int i = (l+r)/2;      // Check middle of remaining subarray
    if (K < A[i]) r = i;       // In left half
    if (K == A[i]) return i;   // Found it
    if (K > A[i]) l = i;       // In right half
  }
  return n; // Search value not in A
}
```

```
int binary (int A[], int n, int K) {
        int l = 0;
        int r = n-1;
        int i ;
        while ( l <= r) {
                i = (l+r)/2;
                if (K  < A[i]) r = i-1;
                if (K == A[i]) return i;
                if (K  > A[i]) l = i+1;
        }
        return n;
}
```

References

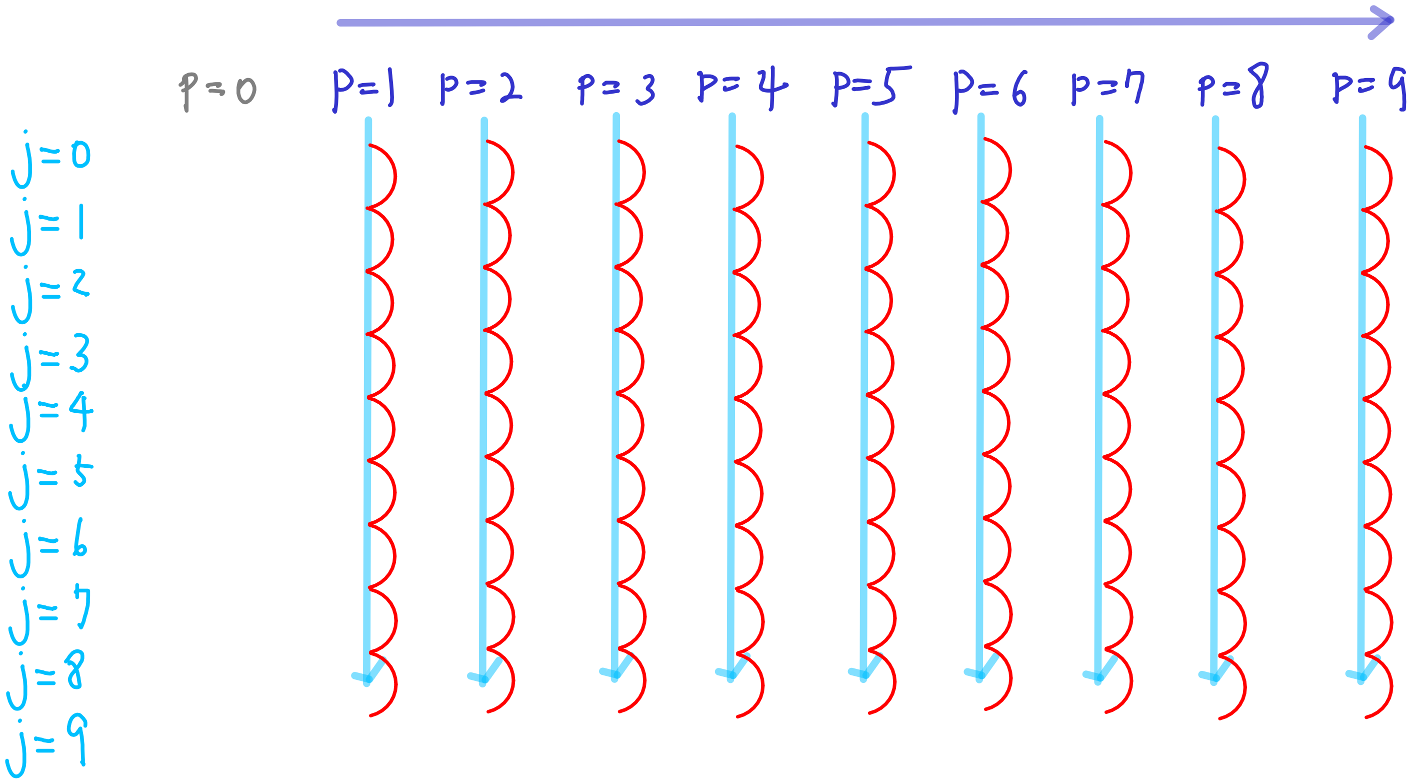
[1] http://en.wikipedia.org/
[2] http://people.cs.vt.edu/shaffer/Book/C++3elatest.pdf

```
#include <stdio.h>

void bubbleSort(int a[], int size) {
  int p, j, tmp;

  for (p=1; p< size; ++p) {
    for (j=0; j< size-1; ++j)
       if ( a[j] > a[j+1] )  {
         tmp = a[j];
         a[j] = a[j+1];
         a[j+1] = tmp;
       }
  }
}


int main(void) {
  int i;
  int a[] = {2, 6, 4, 8, 10, 12, 89, 68, 45, 37};

  bubbleSort(a, 10);



  for (i=0; i<10; ++i)
    printf("a[%d]=%d \n", i, a[i]);

}
```

<

```
   >               <
a[0]=2          a[0]=89
a[1]=4          a[1]=68
a[2]=6          a[2]=45
a[3]=8          a[3]=37
a[4]=10         a[4]=12
a[5]=12         a[5]=10
a[6]=37         a[6]=8
a[7]=45         a[7]=6
a[8]=68         a[8]=4
a[9]=89         a[9]=2
```

```
void bubbleSort(int a[], int size) {
  int p, j, tmp;

  for (p=1; p< size; ++p) {
   for (j=0; j< size-1; ++j)
    if ( a[j] > a[j+1] )  {
      tmp = a[j];
      a[j] = a[j+1];
      a[j+1] = tmp;
    }
  }
}
```


p=0
p=1
p=2
p=3
p=4
p=5
p=6
p=7
p=8
p=9
j=0
j=1
j=2
j=3
j=4
j=5
j=6
j=7
j=8
j=9