**Question1:**

**A:**

1. First I will show my codes of Rand\_select (with linear expected running time). You can see I select the 50th and 80th smallest number in array A.

#include<iostream>

#include<stdlib.h>

#include<time.h>

using namespace std;

int partition(int A[], int p, int q)

{

int d, i, j, x;

x = A[p];

i = p;

for (j = p + 1; j <= q; j++)

{

d = 0;

if (A[j] <= x)

{

i++;

d = A[j];

A[j] = A[i];

A[i] = d;

}

}

d = A[i];

A[i] = A[p];

A[p] = d;

return i;

}

int randomized\_partition(int A[], int p, int q)

{

int i, d, k;

time\_t t;

srand((unsigned)time(&t));

i = rand() % (q - p + 1) + p;

d = A[i];

A[i] = A[p];

A[p] = d;

k = partition(A, p, q);

return k;

}

int rand\_select(int A[], int p, int q, int i)

{

int r,k;

if (p == q)

return A[p];

r = randomized\_partition(A, p, q);

k = r - p + 1;

if (i == k)

return A[r];

else if (i < k)

return rand\_select(A, p, r - 1, i);

else

return rand\_select(A, r + 1, q, i - k);

}

void print\_vector(int v[], int n)

{

int i;

cout << "Vector:";

for (i = 1; i <= n; i++)

cout << " " << v[i];

cout << endl;

}

int main()

{

int A[101];

int i,d,k1,k2;

time\_t t;

srand((unsigned)time(&t));

for (i = 1; i <= 100; i++)

{

A[i] = i;

}

for (i = 1; i <= 100; i++)

{

int num = rand() % 99 + 1;

d = A[i];

A[i] = A[num];

A[num] = d;

}

print\_vector(A, 100);

k1=rand\_select(A, 1, 100, 50);

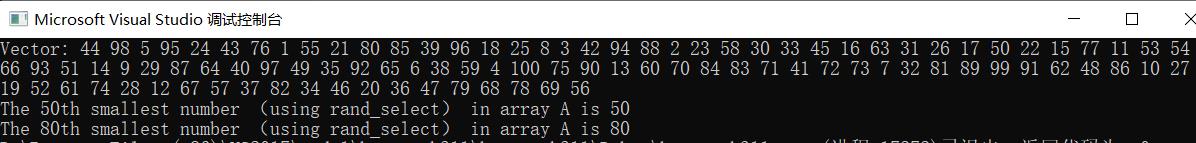
cout << "The 50th smallest number （using rand\_select） in array A is " << k1<<endl;

k2= rand\_select(A, 1, 100, 80);

cout << "The 80th smallest number （using rand\_select） in array A is " << k2;

}

You can see the results as the picture below. I first print the array A as a random permutation.



1. Then I will show my codes of Select (with linear worst-case running time). The input is the same with the first part.

#include<iostream>

#include<stdlib.h>

#include<time.h>

using namespace std;

int partition(int A[], int p, int q)

{

int d, i, j, x;

x = A[p];

i = p;

for (j = p + 1; j <= q; j++)

{

d = 0;

if (A[j] <= x)

{

i++;

d = A[j];

A[j] = A[i];

A[i] = d;

}

}

d = A[i];

A[i] = A[p];

A[p] = d;

return i;

}

void quicksort(int A[], int p, int q)

{

int r;

if (q > p)

{

r = partition(A, p, q);

quicksort(A, p, r - 1);

quicksort(A, r + 1, q);

}

}

// recursively find the median of medians

int find\_median(int A[], int n,int p, int q)

{

if (n == 1)

return A[p];

int i, d, j, k;

int C[21];

d = n / 5;

if (d >= 1)

{

for (i = 0; i <d; i ++)

{

quicksort(A, p+i\*5, p+i\*5+4);

C[i] = A[p+i\*5+2];

}

if (n % 5 != 0)

{

quicksort(A, (p + 5 \* d), q);

C[d] = A[p + 5 \* d + (n % 5 - 1) / 2];

}

}

else

{

quicksort(A, p, q);

C[d] = A[(p + q) / 2];

}

return find\_median(C, d + 1, 0, d);

}

// select the ith smallest number in array A which begins with A[p] and ends with A[q]

int select(int i, int n, int A[], int p, int q)

{

if (n == 1)

return A[p];

int j, k, d, h;

h = 0;

k = find\_median(A,n,p,q);

for (j = p; j <= q; j++)

{

if (k == A[j])

h = j;

}

d = A[h];

A[h] = A[p];

A[p] = d;

d =0;

d=partition(A, p, q);

k = d - p + 1;

if (i == k)

return A[d];

else if (i < k)

return select(i, d-p, A, p, d-1);

else

return select(i-k, q-d, A, d+1, q);

}

void print\_vector(int v[], int n)

{

int i;

cout << "Vector:";

for (i = 1; i <= n; i++)

cout << " " << v[i];

cout << endl;

}

int main()

{

int A[101];

int i, d, k1, k2;

time\_t t;

srand((unsigned)time(&t));

for (i = 1; i <= 100; i++)

{

A[i] = i;

}

for (i = 1; i <= 100; i++)

{

int num = rand() % 99 + 1;

d = A[i];

A[i] = A[num];

A[num] = d;

}

print\_vector(A, 100);

k1 = select(40,100,A,1,100);

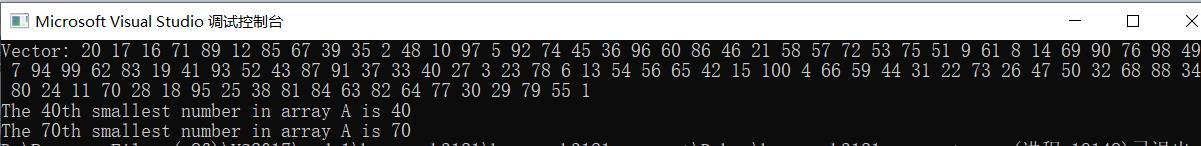
cout << "The 40th smallest number in array A is " << k1<<endl;

k2= select(70, 100, A, 1, 100);

cout << "The 70th smallest number in array A is " << k2;

}

And the result is as following:



**Question2**

**A:** My code of Dynamic Programming of LCS is shown below:

#include<iostream>

using namespace std;

void lcs\_length(char X[], char Y[], int m, int n,int (\*b)[7],int (\*c)[7])

{

int i, j;

for (i = 1; i <= m; i++)

c[i][0] = 0;

for (j = 0; j <= n; j++)

c[0][j] = 0;

for(i=1;i<=m;i++)

for (j = 1; j <= n; j++)

{

if (X[i] == Y[j])

{

c[i][j] = c[i - 1][j - 1] + 1;

b[i][j] = 48;

}

else if (c[i - 1][j] >= c[i][j - 1])

{

c[i][j] = c[i - 1][j];

b[i][j] = 8;

}

else

{

c[i][j] = c[i][j - 1];

b[i][j] = 4;

}

}

}

void print\_lcs(int (\*b)[7], char X[], int i, int j)

{

if ((i == 0) || (j == 0))

return;

if (b[i][j] == 48)

{

print\_lcs(b, X, i - 1, j - 1);

cout << X[i];

}

else if (b[i][j] == 8)

print\_lcs(b, X, i - 1, j);

else

print\_lcs(b, X, i, j-1);

}

int main()

{

char X[9] = { '0','A','B','C','B','D','A','B','\0' };

char Y[8] = { '1','B','D','C','A','B','A','\0' };

int b[8][7];

int c[8][7];

int i;

cout << "The input X is:";

for (i = 1; i < 8; i++)

cout << X[i];

cout << endl;

cout << "The input Y is:";

for (i = 1; i < 7; i++)

cout << Y[i];

cout << endl;

lcs\_length(X, Y, 7, 6, b, c);

cout << "The result of dynamic programming is:";

print\_lcs(b, X, 7, 6);

}

The result is as following. As you see, I use the array X {ABCBDAB} and array Y {BDCABA} as input. But in coding, I add an integer in the beginning of both array for the use of function lcs\_length. And the output {BCBA} fits the bill of dynamic programming.

