1. Average

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void swap(int \*v, int \*s)

{

int temp = \*v;

\*v = \*s;

\*s = temp;

}

void bubble(int arr[], int n)

{

int i, j;

for (i = 0; i < n - 1; i++)

{

for (j = 0; j < n - i - 1; j++)

{

if (arr[j] > arr[j + 1])

swap(&arr[j], &arr[j + 1]);

}

}

}

void selection(int arr[], int n)

{

int i, j, k;

for (i = 0; i < n - 1; i++)

{

k = i;

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

{

if (arr[j] < arr[k])

k = j;

swap(&arr[k], &arr[i]);

}

}

}

void insertion(int arr[], int n)

{

int i, key, j;

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

{

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

int main()

{

clock\_t t;

int i, n = 10000;

srand(time(NULL));

double mt;

int arr[n];

//for input

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

{

arr[i] = rand();

}

//for bubble sort time

t = clock();

bubble(arr, n);

t = clock() - t;

mt = ((double)t) / CLOCKS\_PER\_SEC;

printf("bubblesort() took %f seconds to execute \n", mt);

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

{

arr[i] = rand();

}

//for selection sort time

t = clock();

selection(arr, n);

t = clock() - t;

mt = ((double)t) / CLOCKS\_PER\_SEC;

printf("selectionsort() took %f seconds to execute \n", mt);

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

{

arr[i] = rand();

}

//for insetion sort time

t = clock();

insertion(arr, n);

t = clock() - t;

mt = ((double)t) / CLOCKS\_PER\_SEC;

printf("insetionsort() took %f seconds to execute \n", mt);

return 0;

}

1. Worst case

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void swap(int \*v, int \*s)

{

int temp = \*v;

\*v = \*s;

\*s = temp;

}

void bubble(int arr[], int n)

{

int i, j;

for (i = 0; i < n - 1; i++)

{

for (j = 0; j < n - i - 1; j++)

{

if (arr[j] > arr[j + 1])

swap(&arr[j], &arr[j + 1]);

}

}

}

void selection(int arr[], int n)

{

int i, j, k;

for (i = 0; i < n - 1; i++)

{

k = i;

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

{

if (arr[j] < arr[k])

k = j;

swap(&arr[k], &arr[i]);

}

}

}

void insertion(int arr[], int n)

{

int i, key, j;

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

{

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

int main()

{ //worst complexity

clock\_t t;

int i, j,n = 10000;

srand(time(NULL));

double mt;

int arr[n];

//for input

for (i = 0,j=n; i < n; i++,j--)

{

arr[i] = j+10000;

}

//for bubble sort time

t = clock();

bubble(arr, n);

t = clock() - t;

mt = ((double)t) / CLOCKS\_PER\_SEC;

printf("bubblesort() took %f seconds to execute \n", mt);

for (i = 0,j=n; i < n; i++,j--)

{

arr[i] = j+10000;

}

//for selection sort time

t = clock();

selection(arr, n);

t = clock() - t;

mt = ((double)t) / CLOCKS\_PER\_SEC;

printf("selectionsort() took %f seconds to execute \n", mt);

for (i = 0,j=n; i < n; i++,j--)

{

arr[i] = j+10000;

}

//for insetion sort time

t = clock();

insertion(arr, n);

t = clock() - t;

mt = ((double)t) / CLOCKS\_PER\_SEC;

printf("insetionsort() took %f seconds to execute \n", mt);

return 0;

}

3.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void swap(int \*v, int \*s)

{

int temp = \*v;

\*v = \*s;

\*s = temp;

}

void bubble(int arr[], int n)

{

int i, j;

for (i = 0; i < n - 1; i++)

{

for (j = 0; j < n - i - 1; j++)

{

if (arr[j] > arr[j + 1])

swap(&arr[j], &arr[j + 1]);

}

}

}

void selection(int arr[], int n)

{

int i, j, k;

for (i = 0; i < n - 1; i++)

{

k = i;

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

{

if (arr[j] < arr[k])

k = j;

swap(&arr[k], &arr[i]);

}

}

}

void insertion(int arr[], int n)

{

int i, key, j;

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

{

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

int main()

{ //for best time complexity

clock\_t t;

int i, n = 10000;

srand(time(NULL));

double mt;

int arr[n];

//for input

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

{

arr[i] = i+10000;

}

//for bubble sort time

t = clock();

bubble(arr, n);

t = clock() - t;

mt = ((double)t) / CLOCKS\_PER\_SEC;

printf("bubblesort() took %f seconds to execute \n", mt);

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

{

arr[i] = i+10000;

}

//for selection sort time

t = clock();

selection(arr, n);

t = clock() - t;

mt = ((double)t) / CLOCKS\_PER\_SEC;

printf("selectionsort() took %f seconds to execute \n", mt);

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

{

arr[i] = i+10000;

}

//for insetion sort time

t = clock();

insertion(arr, n);

t = clock() - t;

mt = ((double)t) / CLOCKS\_PER\_SEC;

printf("insetionsort() took %f seconds to execute \n", mt);

return 0;

}

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | sorting Time Complexity | Time Complexity | | |  | |  | Best | Average | worst |  | | Bubble | 0.195 | 0.438 | 0.423 |  | | Selection | 0.395 | 0.573 | 0.424 |  | | Insetion | 0.003 | 0.081 | 0.209 |  | |  |  |  |  |  | |  | |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Screenshot:-

