**1))))))))))))))))))))))))))))))))))))))))))))))))))**

#include <iostream>

#include <string>

using namespace std;

void

print (int n, char arr[], int index)

{

if (index == n)

{

cout << arr << " ";

return;

}

if (arr[index - 1] == '1')

{

arr[index] = '0';

print (n, arr, index + 1);

}

if (arr[index - 1] == '0')

{

arr[index] = '0';

print (n, arr, index + 1);

arr[index] = '1';

print (n, arr, index + 1);

}

}

void

count (int n)

{

if (n <= 0)

return;

char arr[n];

arr[0] = '0';

print (n, arr, 1);

arr[0] = '1';

print (n, arr, 1);

}

int main ()

{

int n = 6;

count (n);

return 0;

}

1. the output for input n = 6 is 000000 000001 000010 000100 000101 001000 001001 001010 010000 010001 010010 010100 010101 100000 100001 100010 100100 100101 101000 101001 101010
2. 6\*21=121(because for each element we do recursion as there are 21 numbers and each of them have 6 elements)
3. for n = 5 = fibonnaci(7) = 13

00000 00001 00010 00100 00101 01000 01001 01010 10000 10001 10010 10100 10101

for n=6 = fibonnaci(8) = 21

000000 000001 000010 000100 000101 001000 001001 001010 010000 010001 010010 010100 010101 100000 100001 100010 100100 100101 101000 101001 101010

for n=7 7= fibonnaci (9) = 34

0000000 0000001 0000010 0000100 0000101 0001000 0001001 0001010 0010000 0010001 0010010 0010100 0010101 0100000 0100001 0100010 0100100 0100101 0101000 0101001 0101010 1000000 1000001 1000010 1000100 1000101 1001000 1001001 1001010 1010000 1010001 1010010 1010100 1010101

d)as every number is like a fibannaci sequence as we see in c) because the numbers that start with 0 are (n-1) and the number that start with ones are (n-2) =>

a[j] = a[j-1] + b[j - 1]

b[j] = a[j-1]

sum = a[j] + b[j]

and the sum is the number of numbers

**2)))))))))))))))**

**with recursion**

#include <iostream>

#include <string>

using namespace std;

void palinomial(char a[], int index, int start) {

if (index <= start) {

cout<<"yes";

return;

}

if (a[start] != a[index]){

cout<< "no";

return;

} else {

return palinomial (a, index-1, start+1);

}

}

int main ()

{

string word = "abda";

int blah = word.length();

char a[]= {'a','b', 'd', 'a'};

palinomial(a, blah-1, 0 );

return 0;

}

**Without recursion**

#include <iostream>

#include <string>

using namespace std;

void palinomial(char c[], int n, int arrsize) {

while(true) {

if(c[n]!=c[arrsize-1-n]) {

cout<<"false";

return;

}

else if(c[n]==c[arrsize-1-n]) {

n++;

}

if(n>=arrsize-1-n) {

cout<<"true";

return;

}

}}

int main () {

char a[]= {'a','b', 'b', 'a'};

int arrsize = 4;

palinomial(a, 0, arrsize );

return 0;

}

**3)))))))))))))))**

class Hardhw

{

void Hardhw(int arr[])

{

int length = arr.length;

for (int i = 0; i < length-1; i++)

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

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

{

int a = arr[j];

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

arr[j+1] = a;

}

}

void printArr(int arr[])

{

int length = arr.length;

for (int i=0; i<length; ++i)

System.out.print(arr[i] + " ");

System.out.println();

}

public static void main(String args[])

{

Hardhw blah = new Hardhw();

int arr[] = {67, 32, 0, 14, 12, 1, 65};

blah.Hardhw(arr);

blah.printArr(arr);

}

}

b)O(n^2)

c) the best case will be already stored array like { 2, 4, 6, 8, 10, 12, 14, 15}

And the worst case will be an array that is storer vice versa {15, 14, 12, 10, 8, 6, 4, 2}

**4)))))))))))))))**

void sort(int array[], int max){

const int maxs = max;

int blah = new int [maxs+1];

int l = maxs;

for(int i = 0; i <max;i++){

blah[i] = 0;

}

for (unsigned int i = 0; i <l; i++){

blah[array[i]]++;

}

int a = 0;

for (unsigned int i = 0 i < l; i++){

for(int j = 0; j<blah[i];j++){

array[a++] = i;

}

}

}

int main() {

int max = 50000

int array[max];

for (int i = 0; i < max; i++){

array[i] = rand() % 50000;

}

for (int j = 0; j < max; j++){

cout<<array[j];

}

sort(array,max);

for (int k = 0; k<max; k++){

cout<<array[k];

}

b)O(n^2)

c)will not be any change

**5)))))))))))))))**

void sort(int array[])

{

int a = array.length;

for (int i=1; i<a; ++i)

{

int blah = array[i];

int j = i-1;

while (j>=0 && array[j] > blah)

{

array[j+1] = array[j];

j = j-1;

}

array[j+1] = blah;

} }

)O(n^2)

c) the best case will be already stored array like { 2, 4, 6, 8, 10, 12, 14, 15}

And the worst case will be an array that is storer vice versa {15, 14, 12, 10, 8, 6, 4, 2}

}

**6)))))))))))))))**

class SelectionSort

{

void sort(int array[])

{

int a = array.length;

for (int i = 0; i < a-1; i++)

{

int min = i;

for (int j = i+1; j < a; j++)

if (array[j] < array[min])

min = j;

int blah = array[min];

array[min] = array[i];

array[i] = blah;

}

}

void printa(int array[])

{

int a = array.length;

for (int i=0; i<a; ++i)

System.out.print(array[i]+" ");

System.out.println();

}

public static void main(String args[])

{

SelectionSort ob = new SelectionSort();

int array[] = {64,25,12,22,11};

ob.sort(array);

System.out.println("Array :");

ob.printa(array);

}

}

b)O(n^2)

c) all arrays will work same way there is no best or worse