1) Rekurencyjny algorytm zmieniający liczbę dziesiętną na binarną + złożoność obliczeniowa.

```
#include <iostream>
#include <stdlib.h>

 using namespace std;

 string DecimalToBinary(long int n){
 return n==0 ? "0" : DecimalToBinary(n/2) + ((n%2)?"1":"0");
 }

 int main(){
 long int t;

 cout<<"Podaj liczbe: ";
 cin>>t;

 cout<< DecimalToBinary(t) <<endl;

 system("pause");
 return 0;
 }
```

Złożoność wynosi O(log2(n)), ponieważ za każdym razem "odrzucamy" połowę liczby. Widać to na przykładzie np ilości wywołań funkcji wyniosą dla liczb np 20,40,80,160,320,640 odpowiednio 5,6,7,8,9,10 wywołań.

2) Czy liczby n liczb całkowitych z przedziały [-2,2] można posortować w czasie O(n) + wyjaśnienie.

Tak liczby mogą zostać posortowane w czasie O(n). Zrealizować to możemy za pomocą algorytmów sortowania przez zliczanie, bądź tez sortowania pozycyjnego.

Dowód dla sortowania przez zliczanie.

Na złożoność obliczeniową tego algorytmu składają się *m* operacji zerowania liczników, następnie *n* operacji zwiększania licznika klucza dla każdego elementu zbioru, wykonanie *m-1* operacji dodawania, przepisywanie elementów ze zbioru wyjściowego do wejściowego i zmniejsza wartości liczników o 1 (ilość wykonanych operacji *n*). Wszystkie te operacje mają kolejno klasy złożoności O(m), O(n), O(m) i O(n). Na czas wykonania ma wpływ wartość *m*, czyli zróżnicowanie elementów do sortowania i *n*, liczba elementów do sortowania. Ostatecznym czasem sortowania jest czas O(m+n).

Żadne z działań nie jest wywoływane wewnątrz siebie (w pętlach), bądź wywoływane rekurencyjnie. Nie pojawia się porównywanie elementów.

3) Zastosuj metodę rekurencji uniwersalnej do rozwiązania następującego równania rekurencyjnego: T(n)=3T(n/3)+n^2.

T(n)=aT(n/b)+f(n)

Rozwiązanie zadania:

$f(n)=n^2$

a=3

b=3

zakładamy epsilon = 1

sprawdzamy $n^{(\log_b a+epsilon)}= n^{(\log_3 3+1)}=n^2$

sprawdzamy czy af(n/b)<= cf(n)

(nie jestem pewien tego) -> wychodzimy od (1/b)*f(n)<=c*f(n) i doprowadzamy do postaci af(n/b)<= cf(n)

(1/3)*n^2 <= c*n^2

(n^2)/3 <= c*n^2

(3*n^2)/9 <= c*n^2

af(n/b)= 3*(n/3)^2

cf(n) = c*n^2

af(n/b)<= cf(n) udowodnione spełnienie, ze dla ¼ <= c < 1

wykresy obu funkcji w pliku PDF

4) Jaka jest złożoność QuickSort w najgorszym przypadku, jeśli elementem osiowym jest mediana? Uzasadnij odpowiedź.

Wybór mediany w algorytmie QuickSort jest przypadkiem optymistycznym złożoności czasowej tego algorytmu, który wynosi w najlepszym wypadku T(n)=n-1+2T*((n-1)/2), a w najgorszym wypadku (dla duzych n) T(n)=n+2T*(n/2).

Złożoności te nie wyniosą O(n)=n^2 ponieważ, wybór mediany nie jest „gwarancją złożoności” która zapewni nam czas wykonania nie większy jak $O(n*\log_2(n))$. Mediana to wartość najczęściej występująca, więc element, który nie zajmuje skrajnych pozycji (przy różnorodności n), dzięki czemu

nie następują podziały nieregularne na części o dużej i małej ilości elementów. Powstają mniej więcej równe części, które są następnie sortowane.

5) Podaj algorytm przekształcenia liczby dziesiętnej na binarną bez zapisywania bitów w tablicy (liście itp.).

```
#include <cstdlib>
#include <iostream>
#include <cmath>

using namespace std;

double b=0, c=0;

int funkcja(int m)
{
    b= b + (m%10)*pow(2,c);
    m=m/10;
    c++;
    if (m==0)
      cout << b << endl;
    else
        funkcja (m);
        }

int main(int argc, char *argv[])
{
    int n;
    cin >> n;
    funkcja(n);
    system("PAUSE");
    return EXIT_SUCCESS;
}
```