**QUYIDAGI MASALALARNING DASTURINING YOZING VA TUZILGAN ALGORITMNI BAHOLANG.**

**1 – Topshiriq**

*Iqbolshoh Ilhomjonov*

1. A va B sonlarning yig’indisini toping va algoritmning vaqt va xotira bo’yicha murakkabligini aniqlang.

#include <iostream>

using namespace std;

int main()

{

    int n, m; 4 , 4 bayt

    cin >> n >> m; 2

    cout << n + m; 1

}

Jami: 3ta

Vaxt : O(1)

Xotira : 8 bayt

3. Oy raqamini berilgan. Kiritilgan oy qaysi faslga tegishli ekanligini chiqaruvchi programma tuzilsin. (Masalan: 2-oy, “qish") algoritmning vaqt va xotira bo’yicha murakkabligini aniqlang.

#include <iostream>

using namespace std;

int main()

{

    int n;

    cin >> n; 1

    switch (n) n

    {

    case 1: 1

    {

        cout << "Qish"; 1

        break;

    }

    case 2:

    {

        cout << "Qish"; 1

        break;

    }

    case 3:

    {

        cout << "Bahor"; 1

        break;

    }

    case 4:

    {

        cout << "Bahor"; 1

        break;

    }

    case 5:

    {

        cout << "Bahor"; 1

        break;

    }

    case 6:

    {

        cout << "Yoz"; 1

        break;

    }

    case 7:

    {

        cout << "Yoz"; 1

        break;

    }

    case 8:

    {

        cout << "Yoz"; 1

        break;

    }

    case 9:

    {

        cout << "Kuz"; 1

        break;

    }

    case 10:

    {

        cout << "Kuz"; 1

        break;

    }

    case 11:

    {

        cout << "Kuz"; 1

        break;

    }

    case 12:

    {

        cout << "Qish"; 1

        break;

    }

    default:

    {

        cout << "Bunday fasl yo'q"; 1

        break;

    }

    }

}

Vaxt : O(1)

Xotira : 4

5. 0 bilan tugaydigan bo’sh bo’lmagan a1, a2…… musbat sonlar ketma ketligi berilgan a1, a1∙a2, a1∙a2∙a3,…..0 ko’rinishidagi ketma-ketlikni hosil qiish dasturini tuzing va algoritm murakkabligini baholang.

#include <iostream>

using namespace std;

int main()

{

    int n, k = 1; 4 , 4 bayt

    cin >> n; 1

    int arr[n];

    for (int i = 1; i <= n; i++) n+1

    {

        cin >> arr[i]; n

    }

    for (int i = 1; i <= n; i++) n+1

    {

        k = 1; 1

        for (int j = 1; j <= i; j++) n\*(n)

        {

            k \*= arr[j]; n\*(n-1)

        }

        cout << k << endl; 1

    }

}

Vaxt : O(n^2)

Xotira : 8 bayt

7. Navbat bilan tanlash orqali ikkita bir o’lchovli massivlarni qo’shish dasturini tuzing va murakabligini baholang.

#include <iostream>

using namespace std;

int main()

{

    int n; 4 bayt

    cin >> n; 1

    int arr1[n], arr2[n], arr3[n];

    for (int i = 0; i < n; i++) n+1

    {

        cin >> arr1[i]; n

    }

    for (int i = 0; i < n; i++) n+1

    {

        cin >> arr2[i]; n

    }

    for (int i = 0; i < n; i++) n+1

    {

        arr3[i] = arr1[i] + arr2[i]; n

    }

    for (int i = 0; i < n; i++) n+1

    {

        cout << arr3[i] << endl; n

    }

    return 0;

}

Vaxt : O(n)

Xotira : 4 bayt

9. N ta butun sonlar ketma-ketligi berilgan. Tartib nomeri o’zining qiymatiga mos kealdigan massiv elementlari yig’indisini hisoblash dasturini tuzing va algoritm murakkabligini baholang.

#include <iostream>

using namespace std;

int main()

{

    int n, k = 0; 4 , 4 bayt

    cin >> n; 1

    int arr[n];

    for (int i = 0; i < n; i++) n+1

    {

        cin >> arr[i]; n

    }

    for (int i = 0; i < n; i++) n+1

    {

        if (arr[i] == i) n

        {

            k += i; n

        }

    }

    cout << "Massiv elementlari indeksiga to'gri keladigan elementlar yig'indisi:" << k; 1

    return 0;

}

Vaxt : O(n)

Xotira : 8 bayt

11.Berilgan natural sonni bo’luvchilari yig’indisini topish dasturini tuzing va algoritm murakkabligini baholang.

#include <iostream>

using namespace std;

int main()

{

    int n, k = 0;

    cin >> n; 1

    for (int i = 1; i <= n; i++) n+1

    {

        if (n % i == 0) n

        {

            k += i; n

        }

    }

    cout << k; 1

}

11. Haqiqiy sonli massivning berilgan sondan kichik barcha elementlari ko’paytmasini toping.

#include <iostream>

using namespace std;

int main()

{

    int n, m, k = 1; 4 , 4 , 4 bayt

    cin >> n >> m; 2

    int arr[n];

    for (int i = 0; i < n; i++) n+1

    {

        cin >> arr[i]; n

    }

    for (int i = 0; i < n; i++) n+1

    {

        if (arr[i] < m) n

        {

            k \*= arr[i]; n

        }

    }

    cout << k; 1

}

Vaxt : O(n)

Xotira : 12 bayt

13.Polinomial bahoga ega bo’lgan algoritmga misol keltiring , dasturini tuzing va murakkabligini baholang

Javob: Agar O( ) bo'lsa bunday algoritmlar polinomial algoritmlar deyiladi. Bunga misol sifatida ichma-ich siklni keltirishimiz mumkin. M x N o’chamli massiv dasturini tuzamiz:

#include <iostream>

using namespace std;

int main()

{

    int n, m; 4 , 4 bayt

    cin >> n >> m; 2

    int arr[n][m];

    for (int i = 0; i < n; i++) n+1

    {

        for (int j = 0; j < m; j++) m+1

        {

            cin >> arr[i][j]; m\*n

        }

    }

}

Vaxt : O (n+m)

Xotira : 8 bayt

15.Tezkor algoritmlarga misollar keltiring.

𝑂(𝑛) va 𝑂(𝑛 log 𝑛) deb baholangan algoritmlar tezkor algoritmlar deb ataladi.

for (i=1; i<=50; i++) 50

s=s+I; 51

Jami:50+51=101

f (101) = O(1); chiziqli murakkablik O(1)

Bu algoritm tezkor algoritmga misol bo’la oladi.

17. Murakkabligi O(n) bahoga ega bo’lgan algoritmga misol keltiring, dasturini tuzing va murakkabligini baholang.

for(i=0; i<=n; i++) n+2

    {

        s=i\*i; n+1

        P=P+i; n+1

    }

Vaxt : O(n)

19. Murakkabligi O(logn) bahoga ega bo’lgan algoritmga misol keltiring, dasturini tuzing va murakkabligini baholang.

 case 1: f(int n) {

      int i;

      for (i = 1; i < n; i=i\*2)

        printf("%d", i);

    }

Vaxt : O(log n)

 case 2  : f(int n) {

      int i;

      for (i = n; i>=1 ; i=i/2)

        printf("%d", i);

    }

Vaxt : O(log n)

21. N butun soni berilgan (n>1). N sonidan katta bo’lgan birinchi Fibonachchi sonini aniqlovchi dastur tuzing va algoritmni vaqt va xotira bo’yich murakkabligini baholang.

#include <bits/stdc++.h>

using namespace std;

int fib(int n)

{

    if (n <= 1) n

        return n; n-0…n-n

    return fib(n - 1) + fib(n - 2); n-0…n-n

}

int main()

{

    int n; 4 bayt

    cin >> n; 1

    cout << fib(n); 1

    getchar();

    return 0;

}

Vaxt : O(n)

Xotira : 4 bayt

23. Berilgan natural son raqamlarini teskari tartibda almashtiring va hosil bo’lgan sonni berilgan son bilan taqqoslang va algoritmni vaqt bo’yicha murakkabligini baholang.

25. Quyidagi keltirilgan qiymatlarni algoritm murakkabligini aniqlang

1= O(1),

n+ 47 = O(n),

+26= O(),

n!= O(n!),

log 5 n+4 = O(log 5 n),

n\*log 2n = O(n\*log 2n),

m+n = O(m+n),

e\*n6 = O(e\*n6) ,

98𝑛 ∗ ln (𝑛) = O(98𝑛 ∗ ln (𝑛))