Binary search

inline int max\_complete(int money\_left)

{

int l = 0, r = k;

while (l < r)

{

int m = (l + r + 1) / 2; //for l = 1 r = 2 case

if (d[m] <= money\_left) l = m; else r = m-1;

}

return c[l];

}

/\*algo to find 100! Or more\*/

#include<stdio.h>

int main()

{

int n,a[200],index,temp,i;

while(scanf("%d",&n))

{

a[0]=1;

index=0;

for(;n>=2;n--)

{

temp=0;

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

{

temp=(a[i]\*n)+temp;

a[i]=temp%10;

temp=temp/10;

}

while(temp>0)

{

a[++index]=temp%10;

temp=temp/10;

}

}

for(i=index;i>=0;i--)

printf("%d",a[i]);

printf("\n");

}

return 0;

}

Compare function

bool compare( data a, data b ) {  
    if( a.income == b.income ) {  
        if( a.height == b.height ) {  
            if( a.weight == b.weight )

              return strlen( a.name ) < strlen( b.name );  
            else return a.weight < b.weight;  
        }else return a.height > b.height;  
    }else return a.income > b.income;  
}

Extended Gcd

int gcdExtended(int a, int b, int \*x, int \*y)

{

    // Base Case

    if (a == 0)

    {

        \*x = 0;

        \*y = 1;

        return b;

    }

    int x1, y1; // To store results of recursive call

    int gcd = gcdExtended(b%a, a, &x1, &y1);

    // Update x and y using results of recursive

    // call

    \*x = y1 - (b/a) \* x1;

    \*y = x1;

    return gcd;

}

Extended Euclid + Inverse Modulo

long long modInverse(long long int a, long long int n)

{

    long long int i = n, v = 0, d = 1;

    while (a>0)

    {

        int t = i/a, x = a;

        a = i % x;

        i = x;

        x = d;

        d = v - t\*x;

        v = x;

    }

    v %= n;

    return v;

}

Bitwise Sieve

#include <bits/stdc++.h>

#define set(x,n) (x[n>>6] |= (1<<((n>>1)&31)))

#define check(x,n) (x[n>>6]&(1<<((n>>1)&31)))

#define MAX 46656

#define LMT 216

using namespace std;

int base[MAX/64];

int primes[5000];

void bitwise\_sieve()

{

unsigned i, j, k;

for(i=3; i<LMT; i+=2)

if(!check(base, i))

for(j=i\*i, k=i<<1; j<MAX; j+=k)

set(base, j);

for(i=3, j=0; i<MAX; i+=2)

if(!check(base, i))

prime[j++]=i;

}

int main()

{

bitwise\_sieve();

for(int i = 0; i < 10; i++)

cout<<primes[i]<<endl;

return 0;

}

Best Seive togather

#include <bits/stdc++.h>

#define ll long long int

using namespace std;

int composite[1000000];

int prime[100000],len=0,len1;

bitset<1010>mnmy;

void func()

{

mnmy.flip();

for(ll i = 3 ; i <= 1000; i+=2)

{

if(mnmy[i])

{

for(ll j = i\*i; j <= 1000; j+=i+i)

mnmy[j]=0;

prime[len++]=i;

}

}

}

int segmented\_seive(int a, int b)

{

len1 = len;

int i, k, cnt = (a<= 2 && 2 <= b) ? 1 : 0;

if(b < 2) return 0;

if(a < 3) a = 3;

if(a%2 == 0) a++;

int sq = sqrt(b);

memset(composite,false,sizeof(composite));

for(i = 0; i < len && prime[i] <= sq; i++)

{

unsigned j = prime[i]\*((a+prime[i]-1)/prime[i]);

if(j%2 == 0) j += prime[i];

for(k = prime[i] << 1; j <= b; j += k)

{

if(j != prime[i]) composite[j-a] = true;

}

}

for(i = 0; i <= b-a; i += 2)

{

if(!composite[i]) prime[len1++]=i+a;

}

}

int main()

{

func();

segmented\_seive(1001,1000000);

return 0;

}

TRIE

#include <bits/stdc++.h>

using namespace std;

struct node

{

bool endmark;

node \*next[26+1];

node()

{

endmark = false;

for(int i = 0; i < 26; i++)

next[i]=NULL;

}

}\*root;

void insert(char \*str)

{

int len = strlen(str);

node \*cur = root;

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

{

int id = str[i]-'a';

if(cur->next[id] == NULL)

cur->next[id]=new node();

cur = cur->next[id];

}

cur->endmark = true;

}

bool search(char \*str)

{

int len = strlen(str);

node \*cur = root;

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

{

int id = str[i]-'a';

if(cur->next[id] == NULL)

return false;

cur = cur->next[id];

}

if(cur->endmark == false)

return false;

return true;

}

void del(node \*n)

{

for(int i = 0; i < 26; i++)

if(cur->next[i])

del(cur->next[i]);

delete(cur);

}

int main()

{

root = new node();

char s[100]="algo";

insert(s);

/\*insert("algea");

insert("algorithm");\*/

if(search(s))

cout<<"YES"<<endl;

return 0;

}

1. Here for case 2 is used for converting a number from x base to desired base   
   calculate:  
   log(x)(N)/ log(x)(b).
2. The number Of digits of N! is **CEILING[log(N) +log (N-1) ... +log(1)]**
3. Ceil – if 3.41 ceil(3.41) = 4
4. Floor-if 3.41 ceil(3.41) = 3

Segment tree

1. #include <bits/stdc++.h>
2. #define ll long long int
3. **using** **namespace** std;
4. ll carry[3\*100003];
5. ll tree[3\*100003];
6. void make\_tree(int node, int fst, int lst)
7. {
8. if(fst==lst)
9. {
10. tree[node]=0;
11. carry[node]=0;
12. return;
13. }
14. int mid = (fst+lst)/2;
15. int left = node<<1;
16. int right = left+1;
17. make\_tree(left,fst,mid);
18. make\_tree(right,mid+1,lst);
19. tree[node]=0;
20. carry[node]=0;
21. }
22. void update(int node,int fst, int lst, int i, int j,int value)
23. {
24. if(fst>=i && lst <= j)
25. {
26. tree[node]+=(lst-fst+1)\*value;
27. carry[node]+=value;
28. return;
29. }
30. if(fst>lst || i>lst || j < fst)
31. return;
32. int mid = (fst+lst)/2;
33. int left = node<<1;
34. int right = left+1;
35. update(left,fst,mid,i,j,value);
36. update(right,mid+1,lst,i,j,value);
37. tree[node]=tree[left]+tree[right]+((lst-fst+1)\*carry[node]);
38. }
39. ll result(int node,int fst,int lst, int i, int j, ll car)
40. {
41. if(fst>=i && lst <= j)
42. return tree[node]+((lst-fst+1)\*car);
44. if(fst> lst||fst>j || lst < i)
45. return 0LL;
47. int mid = (fst+lst)/2;
48. int left = node<<1;
49. int right = left+1;
50. ll p1 = result(left,fst,mid,i,j,carry[node]+car);
51. ll p2 = result(right,mid+1,lst,i,j,carry[node]+car);
52. return p1+p2;
53. }
54. int main()
55. {
56. int n,q,t,cases=0,x,y,z,s;
57. scanf("%d", &t);
58. while(t--)
59. {
60. scanf("%d %d", &n, &q);
61. make\_tree(1,1,n);
62. printf("Case %d:**\n**", ++cases);
63. for(int i = 1; i <= q; i++)
64. {
65. scanf("%d", &s);
66. if(s==0)
67. {
68. scanf("%d %d %d", &x, &y, &z);
69. update(1,0,n-1,x,y,z);
70. }
71. else if(s==1)
72. {
73. scanf("%d %d", &x, &y);
74. ll res = result(1,0,n-1,x,y,0);
75. printf("%lld**\n**",res);
76. }
77. }
78. }
79. return 0;
80. }