Contents

Estructuras de Datos

Segment Tree

```
struct segment_tree{
   int N;
   vector<int> tree;
   segment_tree(int N) : N(N){
      tree.resize(2 * N);
   }
   segment_tree(vector<int>& A){
      N = (int) A.size();
      tree.resize(2 * N);
      build(A, 1, 0, N - 1);
   }
   auto& operator[](size_t i) { return tree[i]; } // this funciton
        works for get element int this position

   // O (n)
   void build(vector<int>& values, int node, int l, int r){
      // if l and r are equal both are leaf node
      // left node = [l, m]
      // m = (l + r) / 2
      // left and right are nodes
      // left interval = [l, m], right intervla = [m + 1, r]
      // after complete fill nodes of left and right, we need to fill
           the [l, r] node
      if(l == r){
         tree[node] = values[l];
         dbg(l, r, node, tree[node]);
         return;
      }
      int m = (l + r) >> 1;
      int left = node + 1;
      int right = node + 2 * (m - l + 1);
      build(values, left, l, m);
      build(values, right, m + 1, r);
      tree[node] = tree[left] + tree[right];
      dbg(l, r, node, tree[node]);
   }

   // O (log N)
   void modify(int pos, int value, int node,  int l,  int r){
      // if l and r are equal, we found our node and update it
      if(l == r){
         tree[node] = value;
         return;
      }
      int m = (l + r) >> 1; // we get the mid
      int left = node + 1;
      int right = node + 2 * (m - l + 1);
      if(pos <= m) modify(pos, value, left, l, m);
      else modify(pos, value, right, m + 1, r);
      tree[node] = tree[left] + tree[right];
   }
   // O( c * log N)
   int query(int ql, int qr, int node, int l, int r){
      if(r < ql || l > qr) return 0;
      if(ql <= l && r <= qr) return tree[node];
      int m = (l + r) >> 1;
      int left = node + 1;
      int right = node + 2 * (m - l + 1);
      int ansL = query(ql, qr, left, l, m);
      int ansR = query(ql, qr, right, m + 1, r);
      return ansL + ansR;
   }

};

void GA(){
   vector<int> A = {2, 3, 1, 7, 5, 8, 3, 2, 5, 1};
   int n = 10;

   segment_tree st(A);
   cout << st.query(1, 5, 1, 0, n - 1) << '\n';
   st.modify(3, 10, 1, 0, n - 1);
   cout << st.query(1, 5, 1, 0, n - 1) << '\n';
   cout << st[1] << '\n';
}

int main(){
   cpu();
   int t = 1;
   //cin >> t;
   while (t--)
   {
      GA();
   }
   return 0;
}
```

Tablas y Cotas

Primos cercanos a 10^n

9941 9949 9967 9973 10007 10009 10037 10039 10061 10067 10069 10079

99961 99971 99989 99991 100003 100019 100043 100049 100057 100069

999959 999961 999979 999983 1000003 1000033 1000037 1000039

9999943 9999971 9999973 9999991 10000019 10000079 10000103 10000121

99999941 99999959 99999971 99999989 100000007 100000037 100000039 100000049

999999893 999999929 999999937 1000000007 1000000009 1000000021 1000000033

Cantidad de primos menores que 10^n

$\pi(10^1) = 4$; $\pi(10^2) = 25$; $\pi(10^3) = 168$; $\pi(10^4) = 1229$; $\pi(10^5) = 9592$; $\pi(10^6) = 78.498$; $\pi(10^7) = 664.579$; $\pi(10^8) = 5.761.455$; $\pi(10^9) = 50.847.534$;
$\pi(10^{10}) = 455.052{,}511$; $\pi(10^{11}) = 4.118.054.813$; $\pi(10^{12}) = 37.607.912.018$

Divisores

Cantidad de divisores (σ_0) para *algunos* $n / \neg\exists n' < n, \sigma_0(n') \geqslant \sigma_0(n)$

$\sigma_0(60) = 12$; $\sigma_0(120) = 16$; $\sigma_0(180) = 18$; $\sigma_0(240) = 20$; $\sigma_0(360) = 24$; $\sigma_0(720) = 30$; $\sigma_0(840) = 32$; $\sigma_0(1260) = 36$; $\sigma_0(1680) = 40$; $\sigma_0(10080) = 72$; $\sigma_0(15120) = 80$; $\sigma_0(50400) = 108$; $\sigma_0(83160) = 128$; $\sigma_0(110880) = 144$; $\sigma_0(498960) = 200$; $\sigma_0(554400) = 216$; $\sigma_0(1081080) = 256$; $\sigma_0(1441440) = 288$ $\sigma_0(4324320) = 384$; $\sigma_0(8648640) = 448$

Suma de divisores (σ_1) para *algunos* $n / \neg\exists n' < n, \sigma_1(n') \geqslant \sigma_1(n)$; $\sigma_1(96) = 252$; $\sigma_1(108) = 280$; $\sigma_1(120) = 360$; $\sigma_1(144) = 403$; $\sigma_1(168) = 480$; $\sigma_1(960) = 3048$; $\sigma_1(1008) = 3224$; $\sigma_1(1080) = 3600$; $\sigma_1(1200) = 3844$; $\sigma_1(4620) = 16128$; $\sigma_1(4680) = 16380$; $\sigma_1(5040) = 19344$; $\sigma_1(5760) = 19890$; $\sigma_1(8820) = 31122$; $\sigma_1(9240) = 34560$; $\sigma_1(10080) = 39312$; $\sigma_1(10920) = 40320$; $\sigma_1(32760) = 131040$; $\sigma_1(35280) = 137826$; $\sigma_1(36960) = 145152$; $\sigma_1(37800) = 148800$; $\sigma_1(60480) = 243840$; $\sigma_1(64680) = 246240$; $\sigma_1(65520) = 270816$; $\sigma_1(70560) = 280098$; $\sigma_1(95760) = 386880$; $\sigma_1(98280) = 403200$; $\sigma_1(100800) = 409448$; $\sigma_1(491400) = 2083200$;
$\sigma_1(498960) = 2160576$; $\sigma_1(514080) = 2177280$; $\sigma_1(982800) = 4305280$; $\sigma_1(997920) = 4390848$; $\sigma_1(1048320) = 4464096$; $\sigma_1(4979520) = 22189440$; $\sigma_1(4989600) = 22686048$; $\sigma_1(5045040) = 23154768$; $\sigma_1(9896040) = 44323200$; $\sigma_1(9959040) = 44553600$; $\sigma_1(9979200) = 45732192$

Factoriales

$0! = 1$	$11! = 39.916.800$
$1! = 1$	$12! = 479.001.600$ ($\in$ `int`)
$2! = 2$	$13! = 6.227.020.800$
$3! = 6$	$14! = 87.178.291.200$
$4! = 24$	$15! = 1.307.674.368.000$
$5! = 120$	$16! = 20.922.789.888.000$
$6! = 720$	$17! = 355.687.428.096.000$
$7! = 5.040$	$18! = 6.402.373.705.728.000$
$8! = 40.320$	$19! = 121.645.100.408.832.000$
$9! = 362.880$	$20! = 2.432.902.008.176.640.000 \in$ `ll`
$10! = 3.628.800$	$21! = 51.090.942.171.709.400.000$

max signed tint = 9.223.372.036.854.775.807
max unsigned tint = 18.446.744.073.709.551.615

University: **Universidad Jose Faustino Sanchez Carrion | UNJFS** Team: **Donde estas CR7** Page 1 of **??**

Consejos

Debugging

- ¿Si n = 0 anda? (similar casos borde tipo n=1, n=2, etc)
- ¿Si hay puntos alineados anda?
- ¿Si es vacío anda?
- ¿Si hay multiejes anda?
- ¿Si no tiene aristas anda?
- ¿Si tiene ciclos anda?
- ¿Si tiene un triángulo anda?
- ¿Los arrays son suficientemente grandes? (siempre denle bastante de más por las dudas, pero tampoco se ceben como para que ya no entre en memoria XD)
- ¿Puede dar integer overflow? (SIEMPRE mirar el integer overflow con MUCHO cuidado)
- ¿Podés dividir por cero en algún caso?
- ¿Estás memorizando la recursión bien?
- ¿El caso base está bien hecho y se llega siempre?
- ¿Están bien puestas las cotas iniciales de la binary / inicialización del acumulador máximo/mínimo?
- ¿Estás inicializando bien antes de cada caso?
- ¿Le copiaste el input dos veces en el archivo de entrada (para ver que de igual y bien las dos veces)? [No aplica cuando viene solo una instancia de input]
- ¿Pasa los ejemplos? [No es joda, Leo se quedo afuera de la mundial por esto]

Hitos de prueba

- **45min** todas las columnas de la tabla llena
- **2h** todos conocen todo
- **3h** reunión estratégica
- **4h** reunión estratégica

University: **Universidad Jose Faustino Sanchez Carrion | UNJFS** Team: **Donde estas CR7**

Page 2 of **??**