|  |  |
| --- | --- |
| **TÌM KIẾM NHỊ PHÂN**  Phần tử có chỉ số lớn nhất **bé hơn bằng**:  ll search(ll f, ll l, ll q)  {      while (f <= l)      {          ll mid = (f + l) / 2;          if (a[mid] > q)              l = mid - 1;          else              f = mid + 1;      }      return l;  }  Phần tử có chỉ số bé nhất **lớn hơn bằng**:  ll search(ll f, ll l, ll q)  {      while (f <= l)      {          ll mid = (f + l) / 2;          if (a[mid] >= q)              l = mid - 1;          else              f = mid + 1;      }      return l + 1;  }  **Chặt đáp án hàm nghịch biến + số thực sai số <=10e-6:**  ld search(ll f, ll l, ll n, ll k)  {      while (f <= l)      {          ll mid = (f + l) / 2;          if (check(1.0 \* mid / 1e6, n, k))              f = mid + 1;          else              l = mid - 1;      }      return 1.0 \* l / 1e6;  }  **Trường hợp dùng được trong thư viện std:**  -> sort();  lower\_bound(a+1,a+n+1,x) - a; -> pos >=x  upper\_bound(a+1,a+n+1,x) - a; -> pos >x  binary\_search(a+1,a+n+1,x) - a;-> true nếu có; else false | **Math**  **(ab) % c**  ll sqr(ll x)  {      return x \* x;  }  ll pw(ll a, ll b, ll MOD)  {      if (b == 0)          return 1 % MOD;      else if (b % 2 == 0)          return sqr(pw(a, b / 2, MOD)) % MOD;      else          return a \* (sqr(pw(a, b / 2, MOD)) % MOD) % MOD;  }  **NcR**  ll printNcR(ll n, ll r)  {      if (n < r)          return 0;      ll p = 1, k = 1;      if (n - r < r)          r = n - r;      if (r != 0)      {          while (r)          {              p \*= n;              k \*= r;              ll m = \_\_gcd(p, k);              p /= m;              k /= m;              n--;              r--;          }      }      else          p = 1;      return p;  } |

|  |  |
| --- | --- |
| **Sàng nguyên tố + xuất vào vector**  const ll num = 100001;  bool kt[num];  vector<ll> b;  void snt(int n)  {      memset(kt, true, sizeof(kt));      kt[1] = false;      for (int i = 2; i <= n; i++)      {          if (kt[i])          {              b.push\_back(i);              int j = 2;              while (j \* i <= n)              {                  kt[j \* i] = false;                  j++;              }          }      }  } | **Sắp xếp lưu vị trí**  bool cmp1(const int &x, const int &y) {  return f[x] < f[y];  }  bool cmp2(const int &x, const int &y) {  return f[x] != f[y] ? f[x] < f[y] : x < y;  }  for (int i=1;i<=n;i++) vt[i] = i;  sort(vt + 1, vt + n + 1 ,cmp1); // cmp1 là lưu vị trí bth, cmp2 là lưu vị trí theo kiểu nếu cùng giá trị a thì vt nào bé hơn thì xếp trước  sort(a+1,a+n+1);  Với cmp1 ta có dùng dùng pair <giá trị, vị trí> rồi sort nó cũng lưu được |
| **Hàm String trong C++**  string(st.rbegin(), st.rend())   * **Đảo xâu ( ưu điểm là thao tác luôn mà không cần biến phụ )** * **st.rbegin() là trả về con trỏ ở cuối** * **st.rend() là trả về con trỏ ở đầu** |  |