**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA: CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI THI GIỮA KỲ**

**THỰC HÀNH CẤU TRÚC DỮ LIỆU & GIẢI THUẬT**

Giảng viên hướng dẫn: Nguyễn Khánh Toàn

Họ và tên sinh viên: Nguyễn Hải Đăng

Mã số sinh viên: 20120049

Lớp: 20CTT1A

Niên khoá: 2021-2022

Thành phố Hồ Chí Minh, ngày 12 tháng 01 năm 2021

**BÀI 1**: (3 điểm)

**Bài này có 2 hàm chính:**

* Hàm addElementAt(List& L, int index, node\* pNew): Hàm có nhiệm vụ định vị node lại cuối (L.tail) sau đó thêm gắn node pNew vào sau vị trí node thứ của danh sách liên kết L (vị trí các node bắt đầu tính từ 1).
* Hàm reList(List& L, int n) : Trong hàm có vòng lặp, biến đếm (), bước nhảy 2 đơn vị.

**Mã giả:**

reList(List& L, int n) {

for (int i = 1; i < n; i += 2) {

node\* pNew = L.tail;

addElementAt(L, i, pNew);

}

}

**Ý tưởng:** sẽ có 1 vòng lặp trong hàm reList(List& L, int n) chạy từ tới khi thì dừng lại và sau mỗi lần lặp thì tăng lên 2 đơn vị. Trong mỗi lần lặp, node\* pNew = L.tail (node cuối hiện tại của danh sách L). Rồi sau đó sẽ gọi hàm addElementAt(L, i, pNew), hàm này sẽ tìm node\* p là node có vị trí thứ i tromg danh sách L, sau đó hàm sẽ định vị lại node cuối sẽ là node liền trước node cuối hiện tại. Sau đó chèn node\* pNew vào sau node\* p đã xác định trước đó.

**Độ phức tạp:**

* Vòng lặp được chạy lần.
* Trong mỗi vòng lặp, độ phức tạp của hàm addElementAt(L, i, pNew) là .
* Độ phức tạp tổng quát của bài là: .

**BÀI 2**: (4 điểm)

**Các hàm chính trong bài;**

* Hàm heapSort(vector<int>& a, vector<int>& b, vector<int>& c, int n): Sắp xếp vector theo thứ tự từ bé đến lớn bằng thuật toán sắp xếp Heapsort, vector và sắp xếp theo cách sắp xếp vector .
* Hàm binarySearch(vector<int>& a, int tmp, int k, int l, int n): Hàm tìm kiếm nhị phân trên vector đã sắp xếp, tìm kiếm vị trí () trên vector thoả sao cho có giá trị nhỏ nhất. Hàm trả về giá trị .
* Hàm createMinus(vector<int> a, vector<int> b) : tạo một vector với các phần tử (). Hàm trả về vector .
* Hàm handle(vector<int>& a, vector<int>& b) : hàm đếm số lượng cặp số sao cho .

**Mã giả:**

handle(vector<int>& a, vector<int>& b) {

vector<int> c = createMinus(a, b);

unsigned long long t = 0;

int n = a.size();

heapSort(a, b, c, n);

for (int i = 0; i < n - 1; i++) {

int b = binarySearch(c, c[i], 0, i + 1, n);

if (b > 0) t += (n - b);

}

return t;

}

**Ý tưởng:**

* Gọi hàm createMinus(a, b) để tạo vector có các phần tử như đã đề cập ở phần các hàm chính. Biến đếm số cặp là , n là số phần tử của vector .
* Sau đó, gọi hàm heapSort(a, b, c, n) để sắp xếp vector tăng dần, vector và được sắp xếp theo cách sắp xếp của vector .
* Trong vòng lặp, biến đếm (), bước nhảy 1 đơn vị.
  + Đặt có giá trị bé nhất () thoả . b có thể trả về giá trị -1 nếu không có phần tử nào thoả.
  + Nếu thì giá trị của biến sẽ tăng đơn vị.
* Trả về giá trị .

**Độ phức tạp:**

* Hàm heapSort() có độ phức tạp là .
* Vòng lặp chạy lần. Trong mỗi lần lặp, hàm binarySearch() có độ phức tạp là .
* Vậy độ phức tạp tổng quát là: .

**BÀI 3:** (3 điểm)

**Các hàm chính trong bài;**

* Hàm heapSort(vector<int>& a, vector<int>& b, vector<int>& c, int n): Sắp xếp vector theo thứ tự từ bé đến lớn bằng thuật toán sắp xếp Heapsort, vector và sắp xếp theo cách sắp xếp vector .
* Hàm binarySearch(vector<int> a, int tmp, int k, int l, int n, int t): Hàm đếm số lượng phần tử trên vector (chỉ số từ 1 tới n – 1) thoả . Hàm trả về giá trị là số lượng phần tử.
* Hàm creTempVector(vector<int> a, int n, vector<int>& b, vector<int>& c): Hàm tạo ra 2 vector và có độ dài với các phần tử thoả:
  + Vector : .
  + Vector : .
  + .
* Hàm countPairs(vector<int>& b, vector<int>& c, int n, int k) : Hàm đếm số cặp thoả yêu cầu đề bài. Trong hàm:
  + Heapsort vector .
  + Heapsort vector .
  + Vòng lặp, biến đếm chạy trong khoảng với bước nhảy 1 đơn vị: đếm số cặp số thoả yêu cầu đề bài. Mỗi cặp thoả mãn thì biến tăng lên 1 đơn vị.
  + Trả về giá trị .

**Mã giả:**

countPairs(vector<int>& b, vector<int>& c, int n, int k)

{

unsigned long long count = 0;

int i;

heapSort(b, n);

heapSort(c, n);

int t = 0;

for (i = 0; i < n - 1; i++)

t = binarySearch(c, b[i], k, 0, n, t);

return t;

}

**Ý tưởng:**

* Trong hàm main(), vector đã được tạo:
* Gọi hàm creTempVector(a, n, b, c) đã tạo vector và thoả yêu cầu của hàm đã được nêu ở phần các hàm chính.
* Gọi hàm countPairs(b, c, n, 0) để đếm số lượng cặp số thoả mãn yêu cầu, trong đó hàm binarySearch(c, b[i], k, i, n) trả về số cặp phần tử thoả mãn đề bài (hiện tại).

**Độ phức tạp:**

* Heapsort có độ phức tạp là .
* Trong vòng lặp, biến đếm chạy từ () với bước nhảy 1 đơn vị, mỗi lần lặp sẽ gọi hàm binarySearch() => độ phức tạp là .
* Độ phức tạp tổng quát là .

**Các nguồn tham khảo**

* <https://www.geeksforgeeks.org/number-of-indices-pair-such-that-element-pair-sum-from-first-array-is-greater-than-second-array/>