ĐỀ THỰC HÀNH CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

Link Code: https://github.com/nguyenkhaan/UIT_Dai_Cuong

Đây là đề thi Thực hành DSA mình tổng hợp lại từ một số ca thi.

Dĩ nhiên, nó không thể giống với đề gốc 100%. Có một số chỗ vì không thể nhớ rõ ràng nên mình đã thực hiện chuẩn hóa lại theo ý mình để đề cho dễ luyện tập nhưng vẫn có thể sát nhất những gì mình còn mang máng. Vì vậy, nội dung trong đây cũng gần giống 80% rồi. Các bạn có thể sử dụng để tham khảo, chuẩn bị cho các bài thi Thực hành IT003.

!! Đây không phải đề chính thức từ trường, nên chỉ nên dùng để **tham khảo và ôn luyện**.

Mình có up source code giải từng đề ở link trên, tuy nhiên, mình không dám chắc có thể AC 100% (tại mình cũng chỉ thi có mỗi đề 1). Do đó, mình khuyến khích mọi người nên tự đọc đề và tự code, tìm ra lời giải cho riêng mình.

Mình xin cảm ơn những bạn đã cho mình xin đề các ca thi để có thể tổng hợp được tài liệu này.

Đóng góp về bài làm, mọi người có thể nhắn tin qua mess của mình để trao đổi

https://www.facebook.com/profile.php?id=61552231689913&locale=vi_VN

Anyway. Chúc mọi người thi tốt

Author: Nguyễn Khả An

Đề 1 (Ca 2 - 28/6/2025)

Câu 1

Một cuộc thi IELTS có nhiều thí sinh tham gia. Mỗi thí sinh được lưu trữ với các thông tin như sau:

- **Mã số thí sinh** (chuỗi ký tự không chứa khoảng trắng)
- **Họ và tên** (chuỗi ký tự có thể chứa khoảng trắng)
- **Năm sinh** (số nguyên)
- **Bốn điểm thành phần** gồm:
    - **Nghe** (số thực)
    - **Nói** (số thực)
    - **Đọc** (số thực)
    - **Viết** (số thực)

Điểm trung bình của một thí sinh được tính theo công thức: **d = (nghe + nói + đọc + viết) / 4**

Điểm trung bình của kì thi sẽ được làm tròn theo nguyên tắc sau:

Phần thập phần từ 0.0 đến dưới 0.25 thì làm tròn thành 0.0: Ví dụ 6.28 thì làm tròn thành 6.0

Phần thập phân từ 0.25 đến dưới 0.75 thì được làm tròn thành 0.5: Ví dụ, 6.62 thì làm tròn thành 6.5

Phần thập phân từ 0.75 trở lên đươc làm tròn thành 1.00: Ví dụ 7.99 thì làm tròn thành 8.0

Yêu cầu: Cài đặt hàm *chon* để lọc ra danh sách các thí sinh có điểm trung bình IELTS d nằm trong khoảng [a , b] cho trước, i.e. d thỏa mãn a ≤ d ≤ b. Thứ tự lọc ra tương ứng với thứ tự sinh viên trong danh sách

Sinh viên chỉ cần cài đặt hàm chon cùng các hàm thành phần tương ứng

Một số hàm cùng cấu trúc lưu trữ struct Thisinh đã được cài đặt sẵn. Sinh viên có thể tham khảo để sử dụng

INPUT	OUTPUT
4 TS001 Nguyen Van A 1999 6.5 6.0 6.0 6.5 TS002 Tran Thi B 2000 7.0 7.0 7.0	TS001 Nguyen Van A 1999 6.25 TS002 Tran Thi B 2000 7.00

7.0 TS003 Le Hoang C 1998 5.0 5.5 5.0 5.5 TS004 Pham Duy D 1997 7.5 7.5 8.0 8.0 6.0 7.5	

Câu 2

Cho một ma trận số nguyên **A** có kích thước **M x N** (*1 ≤ M, N ≤ 600*). Yêu cầu sắp xếp các phần tử trong ma trận sao cho:

- Các phần tử trong **mỗi hàng** đều **tăng dần từ trái sang phải**.
- **Mỗi phần tử** ở hàng trên **lớn hơn mọi phần tử** ở các hàng dưới nó.

INPUT

1. r,c: Số dòng và cột tương ứng của ma trận
2. r x c dòng tiếp theo lần lượt là các phần tử của ma trận A

OUTPUT

Ma trận A sau khi đã sắp xếp thỏa mãn yêu cầu bài toán

INPUT	OUTPUT
3 4 3 9 9 6 10 2 1 4 22 7	4 3 2 1 9 9 7 6 22 16 13 10

16 13	

Sinh viên chỉ cần cài đặt hàm sapxep cùng các hàm thành phần tương ứng

Một số hàm nhapMaTran, xuatMaTran… đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng

Câu 3

Viết chương trình xây dựng **cây nhị phân tìm kiếm (BST)** từ dãy số nguyên được nhập vào, sau đó **tìm và in ra giá trị lớn nhất** được lưu trữ trong cây.

INPUT

1. Một dãy số nguyên được nhập lần lượt từ bàn phím.
2. Kết thúc khi gặp giá trị **-1** (không đưa -1 vào cây).
3. Các số nhập vào đảm bảo có thể tạo thành một cây nhị phân tìm kiếm hợp lệ.

OUTPUT

In ra **giá trị lớn nhất** được lưu trữ trong cây BST vừa tạo.

INPUT	OUTPUT
1 3 2 3 4 5 10 2 4 -1	10

Sinh viên chỉ cần cài đặt hàm maxValue() cùng các hàm thành phần tương ứng

Một số hàm makeNode(), createTree()… cùng cấu trúc lưu trữ BST đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng

Câu 4

Viết chương trình cài đặt **một bảng băm (hash table)** dùng để lưu trữ danh sách các **sinh viên**. Mỗi sinh viên được mô tả bằng các thông tin sau:

- **Mã số sinh viên (MSSV)** – số nguyên (int)
- **Họ và tên** – chuỗi ký tự
- **Giới tính** – ký tự `char: '0'`: Nữ, `'1'`: Nam
- **Năm sinh** – số nguyên
- **Điểm Toán, Lý, Hóa, Anh** – double

Sử dụng **Mã sinh viên (MSSV)** đê làm **khóa.** Hàm băm được định nghĩa như sau: *h(MSSV)=MSSV % M (`M` là **kích thước của bảng băm).***

Khi xảy ra va chạm (collision), sử dụng **phép băm lại** theo công thức:

h'(MSSV,i)=(h(MSSV)+i^2) % M (`i` là số lần thử (probe number) bắt đầu từ 0)

Bảng băm Không giới hạn hệ số tải. KHÔNG LƯU TRÙNG THÔNG TIN. Các **cờ trạng thái** như

`EMPTY`, `DELETED` đã được cài đặt sẵn.

Với các vị trí trống trong bảng băm:

- Nếu là **số nguyên**, giá trị mặc định là `0`.
- Nếu là **ký tự**, giá trị mặc định là `' '` (khoảng trắng).
- Nếu là **chuỗi**, giá trị mặc định là `""` (chuỗi rỗng).

Yêu cầu:

Sinh viên chỉ cần cài đặt hàm createHashTable() cùng các hàm thành phần tương ứng nếu có

Một số hàm inputElement() (Nhập thông tin 1 phần tử), outputElement() (xuất thông tin 1 phần tử) cùng các CTDL lưu trữ Hashtable đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng

INPUT

INPUT	OUTPUT
7 4 10 Nguyen Van A 1 2002 8.0 7.5 8.0 7.0 3 Tran Thi B 0 2001 7.5 7.0 6.5 8.0 17 Le Van C 1 2003 6.0 7.0 6.5 6.5 24 Pham Thi D 0 2000 9.0 8.5 9.0 8.0	17 Le Van C 2003 6.0 7.0 6.5 6.5 0 "" 0 0 0 0 0 0 "" 0 0 0 0 0 10 Nguyen Van A 2002 8.0 7.5 8.0 7.0 3 Tran Thi B 2001 7.5 7.0 6.5 8.0 24 Pham Thi D 2000 9.0 8.5 9.0 8.0 0 "" 0 0 0 0 0

Đề 2 (Ca 2 – 21/6/2025)

Câu 1.

Cho một danh sách liên kết đơn lưu trữ các số nguyên. Hãy viết chương trình sắp xếp lại danh sách sao cho tất cả các số chẵn được đưa lên đầu danh sách, các số lẻ đứng sau, và thứ tự tương đối giữa các node trong từng nhóm (chẵn và lẻ) được giữ nguyên như ban đầu.

INPUT

Một dãy các con số nguyên trong đó: Các số nguyên liên tiếp sẽ được thêm vào danh sách bằng phương pháp thêm cuối, việc thêm vào kết thúc khi gặp -1. -1 không được thêm vào danh sách. Giả sử số nguyên thỏa đk nhập.

OUTPUT

Danh sách liên kết mới sau khi đưa các giá trị chẵn lên đầu DSLK

Lưu ý: Không sử dụng mảng phụ để hỗ trợ.

INPUT	OUTPUT
6 4 -8 2 -5 8 1 4 2 0 7 10 -1	6 4 -8 2 8 4 2 0 10 -5 1 7

Sinh viên chỉ cần cài đặt hàm evenToHead() cùng các hàm thành phần tương ứng, hỗ trợ khác nếu cần Một số hàm createEmptyList(), createList(),… cùng các CTDL lưu trữ DSLK đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng.

Câu 2.

Cho một dãy số nguyên gồm n phần tử và một số nguyên **k** ($1 \le k \le n$). Với mỗi đoạn liên tiếp gồm k phần tử trong dãy, hãy in ra **giá trị nhỏ nhất** trong đoạn đó.

INPUT

Dòng 1: Hai số nguyên n và k — độ dài của dãy và kích thước cửa sổ

Dòng 2: **n số nguyên** — các phần tử của dãy

OUTPUT

In ra **(n - k + 1)** số nguyên — mỗi số là **giá trị nhỏ nhất** trong một cửa sổ k phần tử liên tiếp của dãy, theo thứ tự xuất hiện từ trái sang phải.

INPUT	OUTPUT
8 3	2 2 3 1 1 1

4 2 12 3 5 1 2 8	

Sinh viên chỉ cần cài đặt hàm minInKWindow() cùng các hàm thành phần tương ứng, hỗ trợ khác nếu có

Câu 3.

Cho một **cây nhị phân tìm kiếm (BST)** chứa các số nguyên. Viết chương trình liệt kê tất cả các node có giá trị nằm trong đoạn [m, n], theo thứ tự tăng dần.

INPUT

Số nguyên đầu tiên là gốc của cây. Các số nguyên sẽ được lần lượt thêm vào cây theo thứ tự nhập. Kết thúc khi gặp -1.

Dòng tiếp theo chứa hai số nguyên dương m và n.

OUTPUT

In ra tất cả các giá trị trong cây nằm trong đoạn `[m, n]`, theo thứ tự tăng dần, mỗi giá trị cách nhau một dấu cách. Trong trường hợp cây rỗng thì in ra *Empty Tree.*

INPUT	OUTPUT
10 5 1 7 40 50 -1 5 45	5 7 10 40

Sinh viên chỉ cần cài đặt hàm printNode() cùng các hàm thành phần tương ứng nếu cần

Một số hàm createTree(), makeTreeNode()... đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng

Câu 4.

Cho một chuỗi ký tự **s**, hãy viết chương trình **in ra tất cả các chuỗi con (substring)** khác nhau của chuỗi đó.

- Chuỗi con là một đoạn gồm các ký tự liên tiếp trong chuỗi gốc.
- Các chuỗi con **không được trùng lặp** trong kết quả in ra.
- Các chuỗi con được **in theo thứ tự xuất hiện từ trái sang phải trong chuỗi gốc**

INPUT

Một dòng chứa chuỗi **s,** không chứa kí tự trắng.

OUTPUT

In ra tất cả các chuỗi con **khác nhau**, mỗi chuỗi trên một dòng, theo thứ tự xuất hiện từ trái sang phải

INPUT	OUTPUT
aba	a ab aba b ba

Đề 3 (Ca 3 – 28/6/2025)

Câu 1.

Cho một mảng số nguyên **A** gồm **N phần tử**. Hãy viết chương trình sắp xếp lại mảng theo quy tắc sau:

- **Các số chẵn** được đưa về **đầu mảng** và được sắp xếp **tăng dần**.
- **Các số lẻ** được đưa về **cuối mảng** và được sắp xếp **giảm dần**.

INPUT

Dòng đầu tiên: Số nguyên **N** ($1 \le N \le 10^5$) — số phần tử trong mảng.

Dòng thứ hai: **N số nguyên** — các phần tử của mảng A.

OUTPUT

Mảng A gồm N số nguyên sau khi đã sắp xếp theo quy tắc trên, mỗi số cách nhau một dấu cách.

INPUT	OUTPUT
10 1 4 2 7 6 5 3 8 10 9	2 4 6 8 10 9 7 5 3 1

Sinh viên chỉ cần cài đặt hàm sapxep() cùng các hàm thành phần tương ứng nếu cần

Một số hàm trong template đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng

Câu 2.

Cho một cây nhị phân tìm kiếm (BST) lưu trữ các số nguyên. Hãy viết chương trình thực hiện các yêu cầu sau: **Tính chiều cao** của cây con bên trái và cây con bên phải của nút gốc (root).

- Nếu hai cây con có chiều cao bằng nhau, in ra chiều cao đó.
- Nếu hai cây con có chiều cao khác nhau, in ra tất cả các số âm được lưu trữ trong cây con có chiều cao lớn hơn.

Định nghĩa: **Chiều cao** của một cây là số lượng cạnh trên đường đi dài nhất từ gốc của cây đến một lá.

INPUT

Số nguyên đầu tiên là gốc của cây. Các số nguyên sẽ được lần lượt thêm vào cây theo thứ tự nhập. Kết thúc khi gặp -1.

OUTPUT

Nếu hai cây con có chiều cao bằng nhau: In ra một số duy nhất — **chiều cao** của cây con.

Nếu hai cây con có chiều cao khác nhau: In ra **tất cả các số âm** trong cây con có chiều cao lớn hơn, **mỗi số cách nhau một dấu cách**, **in theo thứ tự tăng dần**.

INPUT	OUTPUT
10 5 15 3 7 13 18 -1	2

Giải thích: Cả cây con trái và phải đều có chiều cao là 2

Sinh viên chỉ cần cài đặt hàm Height(), printNode() cùng các hàm thành phần tương ứng nếu cần

Một số hàm và CTDL trong template đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng

Câu 3.

Công ty **Hoa Sen Xanh** đang triển khai xây dựng một ứng dụng quản lý nhân sự. Mỗi **nhân viên** trong công ty được quản lý với các thông tin sau:

- **Họ và tên**: chuỗi ký tự (`string`)
- **Giới tính**: ký tự (`char`): `'0'`: Nữ, `'1'`: Nam
- **Năm sinh**: số nguyên (`int`)
- **Chỉ số năng lực**: số thực (`double`)

Công ty Hoa sen sử dụng một giá trị gọi là *Chỉ số định mức* để đánh giá mức độ sự chênh lệch năng lực giữa hai nhân viên bất kỳ. Chỉ số định mức giữa hai nhân viên **A** và **B** được tính theo công thức:

$Sai\ lệch = |x - y| + \alpha \times |năm\ sinh\ của\ A - năm\ sinh\ của\ B|$ (x, y lần lượt là chỉ số năng lực của hai nhân viên A và B; α là hằng số bằng *0.917*).

Xây dựng **chương trình quản lý danh sách nhân viên.** Đồng thời, tìm tất cả các cặp nhân viên có **mức độ chênh lệch năng lực nhỏ nhất**.

INPUT

1. Dòng đầu: số nguyên **n** là số lượng nhân viên của công ty
2. Tiếp theo là **n nhóm thông tin**, mỗi nhóm gồm:
   - Họ và tên (1 dòng)
   - Giới tính (0 hoặc 1)
   - Năm sinh (int)
   - Chỉ số năng lực (double)

OUTPUT

In ra tất cả các cặp nhân viên có chỉ số sai lệch nhỏ nhất. Mỗi cặp được in ra theo định dạng:

[Nhân viên A]

[Nhân viên B]

Sai lệch: [giá trị sai lệch làm tròn 3 chữ số thập phân]

Các nhóm thông tin cách nhau bởi một dòng trống

INPUT	OUTPUT
3 Nguyen Van Anh 1 1990 7.5 Tran Thi Binh 0 1992 7.6 Le Van Chung 1 1990 7.4	Nguyen Van Anh 1 1990 7.5 Le Van Chung 1 1990 7.4 0.100

Sinh viên chỉ cần cài đặt hàm printPairs() cùng các hàm thành phần tương ứng nếu cần

Một số hàm và CTDL trong template đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng

Câu 4.

Cài đặt một **bảng băm** để lưu trữ các số tự nhiên nhỏ hơn 10,000, sử dụng phương pháp xử lý đụng độ bằng Linear Probing.

Bảng băm có kích thước tối đa **10,000 phần tử**. Hàm băm được định nghĩa là:

H(key)=key % M (M là kích thước bảng băm).

Hàm băm lại được định nghĩa là:

HF(key , i) = (H(key) + i) % M

Bảng băm **không lưu các khóa trùng nhau**. nghĩa là bảng băm luôn đảm bảo số phần tử được lưu trong bảng băm không quá 70% kích thước của bảng băm.

Sau khi lưu trữ xong các khóa, nhập một số nguyên **k**. Hãy đếm và in ra **số lần duyệt phần tử** trong bảng băm để xóa khóa k khỏi bảng băm Nếu xóa thành công, in ra số lần duyệt. Nếu không tìm thấy khóa **k**, in ra `-1`.

INPUT

Gồm các dòng (giả sử các dòng này luôn thỏa mãn điều kiện đề bài cho)

Dòng đầu: Số nguyên m — Kích thước của bảng băm

Dòng thứ hai: Số nguyên n — số lượng khóa cần chèn vào bảng băm (n ≤ 7000).

Dòng thứ ba: là n số nguyên cách nhau bởi dấu cách — là các số được chèn vào bảng băm)

Dòng thứ tư: Một số nguyên k — khóa cần xóa

OUTPUT

Số lần duyệt phần tử để xóa được khóa k. Nếu không xóa được thì in ra -1

INPUT	OUTPUT
10 7 18 28 19 9981 129 10001 38 129	2

Sinh viên chỉ cần cài đặt hàm createHashtable() cùng các hàm thành phần tương ứng nếu cần thiết

Một số hàm và CTDL trong template đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng