ĐỀ THI: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

(Thời gian 90')
Mã đề CDK54-2010-01
Sinh viên được sử dụng tài liệu

Bài 1.

a) Kích thước của 1 biến kiểu char là 1 Byte, biến kiểu int là 4 byte, kiểu double là 8 byte. Kích thước của 1 con trỏ kiểu char là ? con trỏ kiểu double là ?

Gợi ý: Kích thước của con trỏ không phụ thuộc vào kiểu dữ liệu, mà phụ thuộc vào từng dòng máy. Máy 32 bit thì là 4 byte, máy 64 bit thì là 8 byte. Kiểu int sẽ có kích thước bằng số bit của kiểu máy đó, nếu mà kiểu int là 32 bit tức là đó là máy 32 bit. Như vậy kích thước con trỏ ở đây là 4 byte (32 bit).

b) Đánh giá thời gian thực hiện của thuật toán đệ quy sau theo mô hình O-lớn

Cho ma trận A kích thước $n \times m$, ma trận B kích thước $m \times l$

```
for(i = 0; i < n; i++)
   for( k = 0; k < m; k++)
      for( j = 0; j < l; j++)
          C[i][j] +=  A[i][k] * B[k][j];
```

Hãy đánh giá độ phức tạp của đoạn chương trình trên theo O-lớn

Lệnh cơ sở là lệnh `C[i][j] +=  A[i][k] * B[k][j];`
Có 3 vòng lặp lồng nhau, thời gian thực hiện

$$T(n) = \sum_{i=0}^{n-1}\sum_{k=0}^{m-1}\sum_{j=0}^{l-1} 1 = \sum_{i=0}^{n-1}\sum_{k=0}^{m-1}(l-1+1) =..= n \times m \times l$$

Vậy độ phức tạp cỡ $O(n \times m \times l)$

Bài 2. Trả lời các câu hỏi sau

a) Trong các phương pháp sắp xếp: lựa chọn, chèn, đổi chỗ(nổi bọt), quicksort (sắp xếp nhanh), mergesort (sắp xếp trộn), thì phương pháp nào là phù hợp nhất để sắp xếp trên danh sách liên kết đơn? Giải thích lựa chọn của bạn.

Các thuật toán trên được chia thành 2 loại là cơ bản ($O(n^2)$) và nâng cao ($O(n \log n)$)
Sắp xếp trên danh sách liên kết đơn thì mergesort là phù hợp hơn vì :

- Thời gian trung bình trong trường hợp tồi nhất cỡ $O(n \log n)$
- Cài đặt đơn giản hơn QuickSort

b) Danh sách tên của 1000 sinh viên được lưu trữ trong mảng nhưng không theo 1 thứ tự nào.Hãy nêu những ưu điểm và nhược điểm của phương pháp lưu trữ này
(các tiêu chí đánh giá: bộ nhớ, thời gian tìm kiếm, thêm, xóa)

Ưu điểm:

- Chỉ lưu mỗi tên, không cần lưu thêm thông tin phụ (con trỏ…)
- Có thể truy cập trực tiếp từng phần tử thông qua chỉ số

Nhược điểm

- Có thể lãng phí bộ nhớ nếu không dùng hết
- Vì không cần sắp xếp theo thứ tự nên việc thêm phần tử đơn giản (chỉ cần thêm vào cuối). Tuy nhiên việc tìm kiếm mất nhiều thời gian hơn do chỉ có thể tìm kiếm tuần tự ($O(n)$).
- Thời gian xóa gồm tìm kiếm khóa cần xóa ($O(n)$) và xóa phần tử ($O(1)$ do chỉ cần đổi chỗ với phần tử cuối và giảm số lượng đi 1)

Bài 3.

a) Trong mạng LAN có 1 máy in mạng được sử dụng chung. Các công việc in gửi đến máy được lưu trữ trong 1 hàng đợi, địa chỉ của các công việc được lưu trữ cho đến khi máy sẵn sàng. Công việc mới sẽ được xếp cuối cùng trong hàng đợi. Nêu các lý do tại sao nên dùng hàng đợi cài đặt bằng danh sách liên kết thay vì cài đặt bằng mảng.

Dùng hàng đợi cài đặt bằng danh sách liên kết vì:

- Số lượng công việc in ta không thể biết trước nên dùng danh sách liên kết sẽ tiết kiệm bộ nhớ hơn.
- Ta chỉ lưu trữ địa chỉ của công việc chứ không phải nội dung công việc (nội dung sẽ được để trên máy có yêu cầu in) do đó tiết kiệm được bộ nhớ (do cần phải dùng ít bộ nhớ hơn rất nhiều)

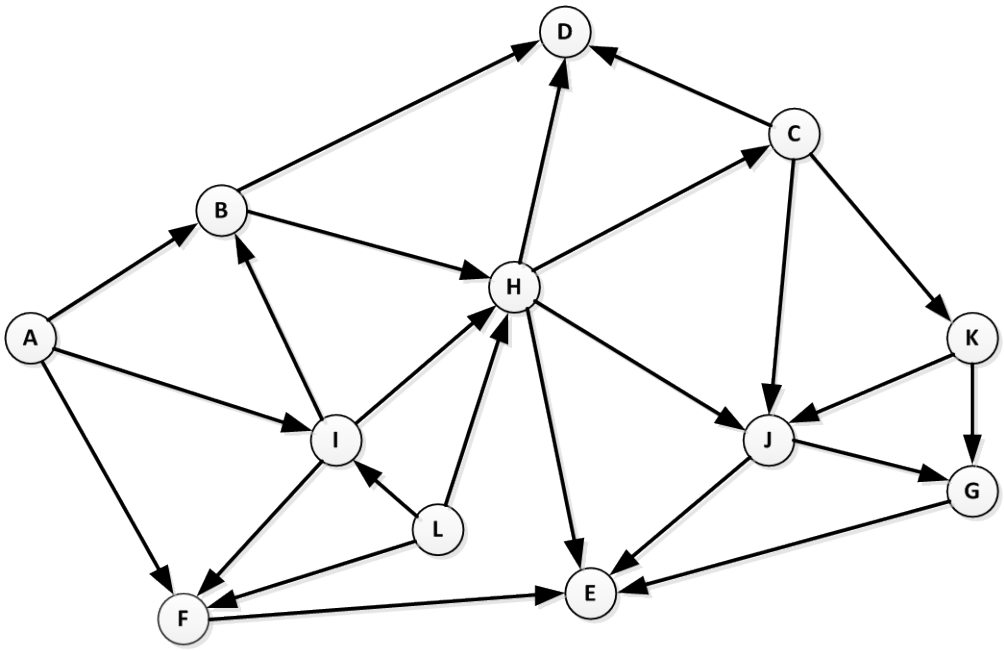
Chú ý: ở đây là hàng đợi nên lấy ra là ở đầu hàng và thêm vào ở cuối hàng, ta không lấy ngẫu nhiên 1 phần tử trong hàng, và sau khi lấy ta cũng không phải dịch các phần tử còn lại.

b) Áp dụng thuật toán chuyển biểu thức dạng trung tố sang dạng hâu tố bằng stack để chuyển biểu thức sau sang dạng hậu tố (cần nêu rõ các bước trung gian trong quá trình tính)

$$3 + 5 \text{ ^ } (12 / 6 + 1) - 7 * 15 / 3 + 6$$

Biểu thức hậu tố tương ứng là :
3 5 12 6 / 1 + ^ + 7 15 * 3 / - 6 +

Bài 4. Cho đồ thị



a. Biểu diễn đồ thị dùng danh sách kề
b. Thực hiện duyệt đồ thị theo chiều sâu (DFS) xuất phát từ đỉnh A, vẽ cây khung thu được

Bài 5. Viết hàm nhận đầu vào là 1 ma trận kề biểu diễn cho 1 đồ thị vô hướng , và số lượng đỉnh trên đồ thị. Hàm kiểm tra xem đồ thị có liên thông hay không. Nếu đồ thị liên thông thì hàm trả về giá trị 1, ngược lại thì hàm trả về giá trị 0.

```
int ktLienThong(int Adj[100][100], int n)
{
   //Thân hàm
}
```

Trong đó n là số lượng đỉnh thực sự trên đồ thị ($0 < n \leq 100$), Adj là ma trận kề lưu trữ đồ thị.

Chú ý : đồ thị liên thông nếu mà giữa hai cặp đỉnh bất kỳ luôn tồn tại đường đi.

```
int ktLienThong(int Adj[100][100], int n)
{
   //Thân hàm
    int Queue[MAX];//Queue de cho BFS
    int QStart,QEnd; //Dau va cuoi queue
    int Color[MAX]; //mau cua cac dinh
    int u,i;

    for(i=0;i<n;i++) Color[i]=-1;
```

```
       //khoi tao queue
      QStart=0;
      QEnd=0;
      //Bat dau tham tu dinh 0
      Queue[0]=0;
      Color[0]=0;

      while(QEnd>=QStart)
      {
            u=Queue[QStart];
            QStart++;

            for(i=0;i<n;i++)
                  if(Color[i]==-1 && Graph[u][i]==1)
                  {
                        QEnd++;
                        Queue[QEnd]=i;
                        Color[i]=0;
                  }
            Color[u]=1;
      }
      //kiem tra xem co ton tai dinh chua tham
      for(i=0;i<n;i++) if(Color[i]==-1) return 0;

      return 1;
   }
```

Thời gian thực hiện $O(n^2)$

Hãy đưa ra đánh giá thời gian thực hiện của thuật toán của bạn theo O-lớn

Chú ý: chương trình sử dụng thuật toán tối ưu hơn sẽ được đánh giá cao hơn!