1. Lý thuyết

Cây nhị phân tìm kiếm:

Khái niệm:

* Mỗi nút là một khoá tìm kiếm.
* Với mỗi cây con, khoá của nút gốc lớn hơn khoá của mọi nút của cây con trái và phải.
* Nếu số lần tìm kiếm các khoá trên cây như nhau thì cấu trúc của cây không quan trọng lắm (cân bằng).
* Nếu số lần tìm kiếm trên các khoá là khác nhau thì khi đó số lần duyệt qua 1 nút có thể tính được: VD: Mức (4) sô lần duyệt : 1+5+3+4+2+3=18.
* Nếu thay đổi cấu trúc của cây thì dẫn đến tổng số lần duyệt có thể sẽ thay đổi.
* Nếu tìm được cấu trúc có tổng số lần duyệt là nhỏ nhất thì chúng ta sẽ có được cấu trúc đem lại hiệu quả nhất (tối ưu).

1. Giải thuật
2. Bái toán:

* Cho mảng A[1,2,…,n] đã sắp xếp theo chiều tăng dần trong đó các phần tử đôi một khác nhau. Mỗi phần tử A[i] có tần sô tìm kiếm f[i] i=1…n.
* Tìm cây nhị phân tối ưu

1. Giải thuật:

* Phân rã

+ Gọi Op(1..n) là số phép so sánh của cây nhị phân tìm kiếm tối ưu của mảng A[1..n]. Nếu A[r] là khóa của nút gốc, ta có:

**Op(1..n) = Op(1..r-1) + Op(r+1..n) + SumF(1..n)**

(SumF(1..n)= f[1]+f[2]+..+f[n])

Vì Op(1..n) là tối ưu nên ta có

**Op(1..n) = min {Op(1..r-1) + Op(r+1..n): r=1..n}+ SumF(1..n)**

**+** Gọi C[i,j] là số phép so sánh của cây nhị phân

tìm kiếm tối ưu cho mảng con A[i..j]

+ Đặt F[i,j] = f[i]+f[i+1]+..+f[j])

+ Ta có

C[i,j] = min{C[i,r-1] + C[r+1,j]: r=i..j} + F[i,j]

+ Bài toán con:

C[i,j] = F[i,j]

+ Tổng hợp:

C[i,j] = min{C[i,r-1]+ C[r+1, j]} + F[i,j]

1. Giả mã:

* Tính F[i,j]:
* Tính C[i,j]:
* Mảng R[i,j]:

1. Độ phức tạp thuật toán:

* Hàm PRECOMPUTE là O(n2)
* Hàm COMPUTECOST là O(n)
* Hàm OPTBINSEARCHTREE là O(n3)

1. Tìm ra cây nhị phân tìm kiếm tối ưu với cây trang Slide 40. Tính kết quả từng bước một