Bài tập CS112: Phân tích thuật toán không đệ quy

Nhóm : 2
Nguyễn Thiện Nhân 23521083
Trần Vinh Khánh 23520726

Bài 1 : Huffman coding

```
//init
For each a in α do:
        Ta = tree containing only one node, labeled "a"
        P(Ta) = pa
        F = {Ta} //invariant: for all T in F, P(T) = ∑_{a∈T} p_α
//main loop
While length(F) >= 2 do
        T1 = tree with smallest P(T)
        T2 = tree with second smallest P(T)
        remove T1 and T2 from F
        T3 = merger of T1 and T2
        // root of T1 and T2 is left, right children of T3
        P(T3) = P(T1) + P(T2)
        add T3 to F
Return F[0]
```

1. **Phân tích và xác định độ phức tạp của thuật toán**

   Thuật toán Huffman Coding gồm 2 thành phần chính

   - **Phần khởi tạo:** với mỗi kí tự a trong bảng chữ cái α, ta tạo một cây T_α chứa duy nhất một nút và nhãn a cùng với chi phí p_a. Do đó độ phức tạp của thao tác này sẽ là số kí tự trong bẳng chữ cái a (đặt là n) nên sẽ có độ phức tạp là $O(n)$
   - **Vòng lặp chính:** với mỗi lần lặp, hay cây có chi phí nhỏ nhất sẽ được chọn và gộp lại thành một cây, rồi sau đó cây mới sẽ được thêm vào danh sách. Với n cây thì sẽ có $n - 1$ bước lặp. Độ phức tạp ở bước này sẽ là $O(n * cost)$ với $cost$ là chi phí để chọn 2 cây nhỏ nhất

   Kết luận : Độ phức tạo của thuật toán là $O(n * cost)$ với $cost$ là chi phí để chọn hai cây nhỏ nhất

2. **Phương pháp để tối ưu thuật toán**

   Để tối ưu thuật toán, ta cần cực tiểu hóa $cost$ (chi phí để chọn 2 cây nhỏ nhất). Nếu làm bình thường chi phí này có thể lên đến $O(n^2)$. Ta có thể sử dụng priorityqueue để làm giảm độ phức tạp này xuống chỉ còn $O(n * log(n))$

Bài 2: Thuật toán Minimum Spanning Tree

Prim

Input: connected undirected graph $G = (V, E)$ in adjacency-list representation and a cost c_e for each edge $e \in E$.
Output: the edges of a minimum spanning tree of G.

// Initialization
$X := \{s\}$ // s is an arbitrarily chosen vertex
$T := \emptyset$ // invariant: the edges in T span X
// Main loop
while there is an edge (v, w) with $v \in X, w \notin X$ **do**
  (v^*, w^*) := a minimum-cost such edge
  add vertex w^* to X
  add edge (v^*, w^*) to T
return T

1. **Mã giả và phân tích độ phức tạp của thuật toán**

```
def Prim(G):
// Chon mot dinh s bat ki trong G
T = {} // Tap canh cua cay khung nho nhat
X = {s} // Tap dinh da duoc chon
while (|X| < |V|): // Lap lai cho den khi tat ca cac dinh duoc chon
    (u, v) = Canh cho chi phi nho nhat voi u thuoc X va v khong thuoc X
    T = T & {(u, v)} // Them canh vao cay khum
    X = X & v // Them dinh v vao X
return T // tra ve cay khung nho nhat
```

Độ phức tạp:

- Khởi tạo với chi phí là $O(1)$
- Vòng lặp sẽ lặp số lần bằng với số đỉnh của đồ thị $|V|$
- Việc chọn ra cạnh có chi phí nhỏ nhất bằng priorityqueue sẽ có độ phức tạp $O(log(|E|))$

Kết luận: độ phức tạp của thuật toán là $O(|V| * log(|E|))$ (đạt yêu cầu đề ra)

Kruskal

Input: connected undirected graph $G = (V, E)$ in adjacency-list representation and a cost c_e for each edge $e \in E$.
Output: the edges of a minimum spanning tree of G.

// Preprocessing
$T := \emptyset$
sort edges of E by cost // e.g., using MergeSort[26]
// Main loop
for each $e \in E$, in nondecreasing order of cost **do**
  if $T \cup \{e\}$ is acyclic **then**
    $T := T \cup \{e\}$
return T

Mã giả và phân tích độ phức tạp của thuật toán

```
def Kruskal(G):
T = {} // Tap canh cua cay khung
sort(E) // Xep xep lai cac canh theo do dai
for (u, v) : E
    if (u, v) khi them vao cay khung khong tao thanh chu trinh
        T = T & (u, v) // them canh (u, v) vao cay khum
return T // tra ve cay khung nho nhat
```

Độ phức tạp:

- Khởi tạo với chi phí là $O(1)$
- Sắp xếp các cạnh sẽ có độ phức tạp là $O(|E|log(|E|))$
- Duyệt qua tất cả các cạnh $O(|E|)$, và kiểm tra chu trình bằng thuật toán DSU với độ phức tạp là $O(log(|E|))$

Kết luận: độ phức tạp của thuật toán là $O(|E| * log(|E|)$ (đạt yêu cầu đề ra)

Hết