Group 10's homework solution

Solution

Subtask 1 ($n \leq 18$)

Với n nhỏ, ta có thể thử tất cả các tập con nhân viên để sa thải (trừ nhân viên 1), tức 2^{n-1} khả năng.

Độ phức tạp thời gian là $\Theta(2^n * n)$.

Subtask 2.1 ($n \leq 369, m \leq 369$)

Gọi $dp[u][c]$ là số nhân viên giữ lại tốt thiểu trong cây con gốc u mà có chỉ số phức tạp bằng c.

Ta sử dụng dp knapsack, để gộp từ đỉnh v trong cây con gốc u lên u, ta mất chi phí $\Theta(m^2)$.

Do đó, độ phức tạp thời gian của thuật toán là $\Theta(n \cdot m^2)$.

Subtask 2 ($n \leq 676767, m \leq 369$)

Mỗi nhân viên u có khối lượng công việc w_u, và số lượng nhân viên trực tiếp dưới quyền u là s_u và chỉ số phức tạp ban đầu

$$C_u = w_u + s_u$$

Việc sa thải một nhân viên v sẽ chuyển toàn bộ công việc w_v và các nhân viên trực tiếp dưới quyền v lên quản lý trực tiếp của v là u. Điều này làm thay đổi chỉ số phức tạp của như sau:

- w_u tăng thêm w_v.
- s_u giảm 1 nhưng tăng thêm s_v.

$\rightarrow$ Tổng tăng chỉ số phức tạp của u là $\Delta_v = w_v + s_v - 1 = C_v - 1$.

Ta định nghĩa lại hàm quy hoạch động $dp[u] = (k_u, c_u)$, trong đó: - k_u: số lượng nhân viên sa thải tối đa trong cây con gốc u (bao gồm u). - c_u: chỉ số phức tạp tối thiểu của u sau khi tối ưu hóa cây con (để đạt k_u). - vì sao thì ta sẽ chứng minh ở dưới

Nếu u là lá, thì $k_u = 0$, $c_u = w_u$.

Với u có các con $v_1, v_2, \ldots, v_d$:

- Ban đầu, chỉ số phức tạp của u là $c_u = w_u + d$.
- Nếu không sa thải ai là con của u thì số người sa thải nhiều nhất bằng $\sum k_{v_i}$.
- Nếu sa thải một con v_i, thì chỉ số phức tạp u tăng thêm $\Delta_{v_i} = c_{v_i} - 1$, và số người nhiều nhất sa thải tăng lên 1.

$\rightarrow$ Bài toán trở thành chọn tập con các v_i nhiều nhất, sao cho $c_u + \sum \Delta \leq m$.

Sử dụng Quy hoạch động knapsack ta định nghĩa $f[j]$ = số lượng nhiều nhất nhân viên trực tiếp của u bị sa thải sao cho có tổng chỉ số phức tạp $\leq j$.

Chi phí tính toán tính u và một nhân viên v của u là $\Theta(m)$.

Do đó độ phức tạp thuật toán là $\Theta(n \cdot m)$

Subtask 3 (Không có ràng buộc gì thêm)

Ta nhận thấy khi sa thải một cây con v_i thì số người ít nhất cần giữ lại giảm đi một lượng cố định bằng 1.

Từ nhận xét trên, thay vì ta sử dụng quy hoạch động knapsack, ta có thể tham lam chọn nhiều nhất số v_i trong u bằng cách chọn từ nhỏ đến lớn tăng dần theo c_{v_i}.

Chi phí để sắp xếp tất cả đỉnh $\Theta(\sum_u deg(u) \log deg(u)) \leq \Theta(n \log n)$ (với $deg(u)$ là nhân viên trực tiếp của u và $deg(u) \leq n-1$).

Do đó độ phức tạp của thuật toán là $\Theta(n \log n)$.

Chứng minh tính đúng đắn

Ta cần chứng minh vì sao chỉ cần lưu dp$[u] = (k_u, c_u)$ là đủ?

- Giả sử ta có các cặp $(removed_i, value_i)$ cho đứa con thứ i với giá trị $removed_v$ là lớn nhất.
- Và giả sử rằng k đỉnh bị xóa trong nghiệm của ta là $v_1, v_2, \dots, v_k$, ta biết rằng $value_1, value_2, \dots, value_k$ là k giá trị nhỏ nhất.
- Giả sử tồn tại một nghiệm tối ưu khác trong đó tồn tại i sao cho $removed_i^\sigma < removed_i$:
- Gọi C là phần chênh lệch trong giá trị mới, tức là $value_i^\sigma = value_i - C$. Ta biết rằng $C \leq value_i$ vì $0 \leq value_i$.
  - Khi $1 \leq i \leq k$, để đạt cùng tổng số đỉnh bị xóa, ta phải tìm một $value_j \leq C$ với $j > i$. Điều này chỉ có thể xảy ra nếu $value_i \leq value_j = C$, do đó nghiệm mới sẽ không bao giờ tốt hơn nghiệm ban đầu.
  - Khi $k < i$, ta có $total + value_i - 1 > m$. Để đạt cùng tổng số đỉnh bị xóa, điều này chỉ có thể thực hiện nếu $total + value_i^\sigma - 1 \leq m$, nhưng vì $value_i^\sigma - 1 \geq 0$, nên nghiệm đó cũng sẽ không tốt hơn $total$.
- Do đó, mỗi khi ta "hoàn tác" việc xóa một nốt trong cây con, nghiệm sẽ không bao giờ tối ưu hơn nghiệm hiện tại của gốc.

- Vì vậy, để xây dựng được kết quả ưu này, mỗi cặp $(removed_i, value_i)$ cũng phải là tối ưu.