**Bài 1: CAKE**

Từ yêu cầu của đề là phải làm giảm độ chênh lệch giữa tổng cân nặng bánh của người i và người j (giảm |sum\_i – sum\_j| ) ta có thể nhìn ra là người có tổng cân nặng lớn hơn sẽ phải đổi một cái bánh nặng hơn với một cái bánh nhẹ hơn của người còn lại, nói cách khác, w\_i > w\_j nếu sum\_i > sum\_j. Vì vậy ta sẽ sắp xếp mảng w lại cho dễ xử lý (sắp xếp mảng p theo mảng w).

Tiếp theo ta cần nhìn ra thêm một vấn đề nữa. Gọi d\_i là phần còn lại của sum\_(p\_i) khi không tính cân nặng w\_i. Nhận xét rằng, sum\_(p\_i) = d\_i + w\_i và sum\_(p\_j) = d\_j + w\_j, từ đó suy ra |sum\_i – sum\_j| = |d\_i + w\_i – d\_j – w\_j|. Từ công thức trên ta có thể thấy, nếu d\_i = d\_j thì chênh lệch không đổi (|w\_i – w\_j| = |w\_j – w\_i|). Vậy chúng ta không thể trao đổi 2 cái bánh i và j nếu nó rơi vào trường hợp đó.

Một trường hợp nữa, ta có sum\_(p\_i) = d\_i + w\_i, với j < i, nếu d\_j > d\_i thì sau khi hoán đổi 2 cái bánh i và bánh j rõ ràng sum\_(p\_j) sẽ lớn hơn sum\_(p\_i). Vì nếu d\_j > d\_i, cộng với việc vốn dĩ w\_i > w\_j thì khi hoán đổi, sum\_(p\_j) = d\_j + w\_i, hai cái lớn hơn cộng vào nhau sẽ tạo ra một cái tổng lớn hơn cái tổng kia. Và không cần xét đến độ lớn ban đầu của sum\_(p\_i) và sum\_(p\_j), ta vẫn hoàn toàn có thể nhận thấy được, nếu d\_j > d\_i thì (d\_j + w\_i) > (d\_i + w\_i), (d\_j + w\_j) > (d\_i + w\_j). Vậy nếu d\_j > d\_i, việc tráo đổi 2 cái bánh i và j chỉ làm cho chênh lệch của ban đầu trở nên lớn hơn. Vậy ta cũng loại bỏ trường hợp d\_j > d\_i.

Tóm tắt lại, ta sẽ sắp xếp lại mảng w, sau đó tính mảng sum và mảng d. Nhiệm vụ cuối cùng là với mỗi i ta sẽ đếm số lượng j (j < i) mà có d\_j < d\_i. Công việc đếm có thể tối ưu bằng cách sử dụng cấu trúc dữ liệu IT hoặc BIT trên khoảng giá trị của mảng d, vì số có thể rất lớn nên sau khi tính mảng d hãy nén lại.

Ngoài ra, còn một vấn đề phát sinh là vì nếu w\_i = w\_j thì hoán đổi cũng không được gì. Vì vậy ta chỉ cần kiểm tra nếu w\_i > w\_(i-1) thì ta mới update một lượt những cái bánh có cùng cân nặng.

Và cũng một trường hợp xảy ra là nếu ta lỡ tính luôn việc hoán đổi 2 cái bánh của cùng một người thì sao? Rất may là ta đã vô tình loại trừ được trường hợp đó. Giả sử 2 cái bánh i và j (j < i) là của cùng một người, thì ta thấy sum\_(p\_i) = sum\_(p\_j) = d\_i + w\_i = d\_j + w\_j, ta có w\_i >= w\_j suy ra d\_j >= d\_i, mà nó lại rơi vào 2 trường hợp ta đã loại trừ ở trên, vậy ta đã vô tình xử lý được trường hợp này.

Code C++:<https://ideone.com/77pp9c>

Code Pascal: <https://ideone.com/usT9sS>

**Bài 3:**

Đề bài yêu cầu tìm đường đi từ u đến v, từ đó xác định đỉnh cần tới tiếp theo. Vậy ta có một nhận xét đơn giản nhất: nếu u và v không cùng một miền liên thông, ta sẽ không tìm được đường đi (in ra -1).

Việc còn lại cần làm là tìm đường đi nối từ u đến v. Cho k truy vấn (k <= 2e5), vậy suy ra trong mỗi truy vấn ta chỉ được xử lý trong O(1) hoặc O(log2). Việc tìm ra toàn bộ đường đi từ u đến v rõ ràng sẽ không thoả được độ phức tạp ở trên. Nhưng ta không cần thiết phải tìm ra toàn bộ đường đi, ta chỉ cần tìm ra đỉnh kề u mà nằm trên đường đi đó thôi.

Giả sử ta xây dựng được một cây LCA có gốc tại một đỉnh nào đó, sẽ có 2 trường hợp xảy ra:

1. u và v là con ở 2 nhánh khác nhau của một đỉnh x nào đó, nói cách khác, LCA(u,v) = x. Vậy ta có thể thấy đỉnh tiếp theo cần đi tới là đỉnh cha trực tiếp của u.
2. u hoặc v là tổ tiên của đỉnh còn lại. Vậy ta chỉ cần xét, nếu độ dài từ đỉnh tới u (h[u]) dài hơn độ dài từ đỉnh tới v (h[v]), ta cũng chỉ cần in ra đỉnh cha trực tiếp của u. Mặt khác, nếu h[v] > h[u], tức u là tổ tiên của v, vậy ta cần tìm ra tổ tiên ngay dưới u, nói cách khác là tìm tổ tiên x của v sao cho h[x] = h[u]+1 (phải làm như vậy vì ta cần tìm đỉnh tiếp theo kề u, mà u thì có nhiều con, trong khi v thì chỉ có thể xác định được duy nhất một tổ tiên x như vậy, như vậy việc tìm từ v lên cho trường hợp này là hợp lý hơn).

Nhận xét thấy rằng, vì đồ thị đã cho là một cây, mà trên cây thì chỉ có một đường đi duy nhất từ u đến v. Vậy ta có thể chọn bất kỳ đỉnh nào trong một miền liên thông để làm gốc cho cây LCA ta dựng ở miền liên thông đó, vì việc dựng cây LCA tốn O(NlogN) nên ta chỉ nên chọn cố định một đỉnh làm gốc cho tất cả truy vấn (tất nhiên cũng không ai ngu đến mức cứ mỗi truy vấn lại chọn lại gốc :v).

Code Pascal: <https://ideone.com/AJlPSd>

Code C++: updating...