Bài 1. Dãy con DAYCON.*

- **Subtask 1:** Với n=1, dãy số này chỉ có một số thì chỉ có một cặp số (1,1), ta chỉ cần kiểm tra xem số này có phải là số fibonacci hay không, nếu có in ra 1, ngược lại in ra 0.
- **Subtask 2 và Subtask 3:** Gọi mảng prefix là mảng cộng dồn từ 1 → n

Với mỗi i duyệt từ Q → N, vì độ dài đoạn nằm trong đoạn P →Q nên với mỗi i ta xét các prefix trong đoạn i – Q +1 → i– P +1, duyệt j từ i – Q +1 → i– P +1 nếu prefix[i]- prefix[j] là số fibonacci thì kết quả tăng 1.

Độ phức tạp là $O(n^2)$

- **Subtask 4:** Nhận xét dãy có 10^5 phần tử với giá trị lớn nhất của các phần tử là 10^6 thì tổng dãy lớn nhất là 10^{11}, mà số fibonacci thứ 60 là 956722026041, lớn hơn 10^{11}. Do đó ta có thể lật ngược bài toán.

Xét với mỗi số fibonacci thì ta đếm có bao nhiêu đoạn có tổng chính bằng số fibonacci này có độ dài ≥ P và ≤ Q.

Độ phức tạp: O(60*n)

Bài 2. Mèo đuổi chuột MEOCHUOT.*

Nhận xét: Tại một thời điểm bất kì xét đường đi từ Mèo (T) đến Chuột (J) thì thấy rằng tất cả các vị trí bên từ giữa đến J thì Chuột đều có thể chạy đến mà không bị bắt. Do đó để kéo dài trò chơi thì J nên chọn vị trí xa vị trí giữa nhất tính về nửa bên phải của mình (như hình).

Như nhận xét, ta thấy để giải quyết mỗi truy vấn, ta cần giải quyết 2 bài toán nhỏ:

- Bài toán 1: Tìm điểm ở giữa nhanh nhất (cần tính khoảng cách $T \to J$, cần xác định nhanh đỉnh nào ở giữa,...), đây liên quan đến LCA. Độ phức tạp $O(\log(N))$, nhưng tiền xử lí cho toàn bài trong $O(N.\log(N))$.

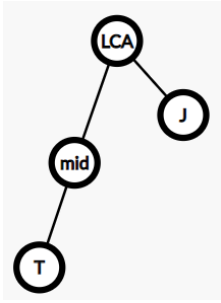
Ta sẽ tính khoảng cách $T \to J$

$$dist = depth[t] + depth[j] - depth[lca(t,j)] * 2$$

Khoảng cách từ $T \to mid$: $distT = dist/2$

Khoảng cách từ $J \to mid$: $distJ = (dist - 1)/2$

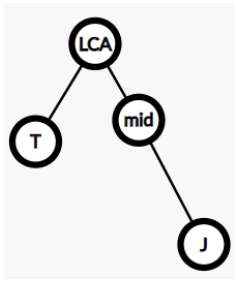
- ***Bài toán 2:*** Xét trong nửa bên phải đường đi phía J, cần tìm điểm xa điểm giữa nhất, đây là bài toán quy hoạch động trên cây. Độ phức tạp mỗi truy vấn $O(1)$, nhưng tiền xử lí cho toàn bài trong $O(N)$.



Ta sẽ dùng DFS để duyệt cây, cập nhật khoảng cách xa nhất từ dưới lên. Với mỗi đỉnh u cần có được thông tin đỉnh xa nhất trong nhánh DFS gốc u, và thông tin đỉnh xa nhất từ u ra ngoài nhánh DFS gốc u.

Trường hợp 1: $depth[T] > depth[J]$

Kết quả sẽ là khoảng cách từ $T \rightarrow mid$ cộng với khoảng cách xa nhất từ mid ra ngoài cây con DFS gốc mid



$$res1 = distT + up[mid]$$

Trường hợp 2: depth[T] ≤ depth[J]

Kết quả sẽ là khoảng cách từ $J \rightarrow mid$ cộng với khoảng cách xa nhất từ mid tới đỉnh trong cây con DFS gốc mid

$$res2 = distT + down[mid]$$

Độ phức tạp thuật toán: $O(N.\log(N) + Q.\log(N))$

Bài 3. Gấu trúc

Subtask 1: Duyệt tất cả các hoán vị rồi tính kết quả trong $O(\text{N!})$

Subtask 2:

Xét một cạnh $e = (v, w)$ trên cây. Đặt n_v, n_w lần lượt là kích thước của hai cây con khi cắt cây theo cạnh e. Số lượng gấu đi qua cạnh này (gọi là s_e) sẽ không vượt quá $2\min\{n_v, n_w\}$.

Tổng khoảng cách được định nghĩa trong bài toán là tổng của các s_e, do đó tổng này không vượt quá $\sum 2\min\{n_v, n_w\}$. Trên thực tế, luôn có thể đạt được con số lớn nhất này.

Xét hai trường hợp:

- Cây có hai trọng tâm:

Gọi hai trọng tâm là $c_1$và c_2. Giá trị lớn nhất có thể đạt được khi tất cả Gấu đi qua cạnh giữa c_1 và c_2. Có $(N/2)! \times (N/2)!$ cách.

- Cây chỉ có một trọng tâm:

Gọi trọng tâm này là c. Giá trị lớn nhất có thể đạt được khi tất cả gấu đi qua đỉnh c.

Đặt $T_1, T_2, \dots, T_K$ là các cây con thu được khi xoá đỉnh c khỏi cây gốc. Ta cần đếm số lượng hoán vị mà với mỗi i, không con gấu nào trong T_i đi đến một đỉnh trong T_i.

Sự khác biệt giữa subtask 2 và 3 nằm ở cách tính kết quả này.

Đối với subtask 2, chúng ta có thể thực hiện một thuật toán DP $O(N^3)$ như sau:

Đặt tất cả các đỉnh vào một hàng từ trái sang phải từ nhóm (cây con) 1 đến nhóm K, các đỉnh cùng một nhóm thì nằm kề nhau.

$f[i][j] =$ số cách để tất cả các con gấu từ nhóm 1 đến nhóm i di chuyển tới một đỉnh thuộc nhóm khác và còn dư ra j con gấu (trong i nhóm đầu tiên) sẽ di chuyển tới các nhóm lớn hơn i => Điều này cũng có nghĩa là cần j con gấu ở các nhóm lớn hơn i di chuyển tới các nhóm bé hơn hoặc bằng i (định luật bảo toàn … số lượng)

Kết quả là $f[K][0]$

Cách tính: khi đi từ nhóm i đi tới nhóm $i+1$, giả sử đã tính được $f[i][j]$, xét số lượng gấu trong nhóm hiện tại là $|T_i|$. Những con gấu trong nhóm $i+1$ này có hai lựa chọn, một là di chuyển tới những vị trí còn trống ở các nhóm trước đó (j vị trí), còn lại là vào hàng chờ và di chuyển tới những nhóm phía sau.

- ⇨ Duyệt số lượng gấu trong nhóm $i+1$ sẽ đi vào các nhóm phía trước, từ đó thực hiện cập nhật trạng thái một cách thích hợp (coi như bài tập cho các bạn).

Subtask 3:

Thay vì sử dụng DP ngay từ đầu để tính kết quả thì chúng ta sử dụng bao hàm loại trừ:

Đặt $T := T_1 \cup T_2 \cup \cdots \cup T_K$ (i.e., tất cả các đỉnh trừ đỉnh c). Kết quả là $\sum_{S \subseteq T} (-1)^{|S|} f_S$, trong đó f_S là số lượng hoán vị của các con gấu mà mỗi con gấu trong S đi tới một đỉnh trong cùng nhóm.

Với mỗi k, ta cần tính tổng của f_S cho mọi S mà $|S| = k$.

Gọi g_k là số lượng cách để chọn k con gấu và đích đến của chúng, sao cho tất cả con gấu được chọn sẽ đi đến một đỉnh cùng nhóm. Cụ thể, g_k là số lượng hai bộ k đỉnh $(s_1, \dots, s_k), (t_1, \dots, t_k)$ mà:

- $s_1 < s_2 < \cdots$ (thứ tự của các con gấu được chọn không quan trọng)
- Với mỗi i, s_i và t_i ở trong cùng cây con (và chúng không phải c).
- t_i là đôi một phân biệt

Vì chúng ta có thể tự do chọn đích đến cho những con gấu chưa được chọn, tổng của f_S cho tất cả S mà $|S| = k$ là $g_k(N-k)$!. Do đó, kết quả là $\sum_{k=0}^{N-1} g_k(N-k)!\,(-1)^k$. Vì g_k có thể dễ dàng tính trong $O(N^2)$ bằng phương pháp DP đơn giản, chúng ta cũng có thể tính được kết quả trong $O(N^2)$.