Bài 1 Cặp đôi
Subtask 1: Duyệt mọi cách ghép cặp có thể và kiểm tra chênh lệch nhỏ nhất

Subtask 2:
- Chia nhị phân chênh lệch nhỏ nhất (gọi là X)
- Xây dựng đồ thị 2 phía với C[i, j] = 1 nếu chênh lệch giữa Bi và Gj không quá X
- Kiểm tra bộ ghép cực đại có đạt độ lớn là K không, nếu không đạt thì tăng X, nếu đạt thì giảm X
- Độ phức tạp log(10^9) * độ phức tạp tìm cặp ghép

Subtask 3:
Tương tự như subtask 2, nhưng cần tối ưu đoạn tìm cặp ghép. Một cách làm khác là:
- Sắp xếp mảng B và G.
- Chia nhị phân chênh lệch nhỏ nhất
- Quy hoạch động F[i, j] là số cặp nhiều nhất khi xét i bạn nam đầu tiên, và j bạn nữ đầu tiên:
    - F[i, j] = F[i - 1, j - 1] + 1 nếu |Bi - Gj| <= X
    - F[i, j] = max(F[i - 1, j], F[i, j - 1]) nếu ngược lại.
- Độ phức tạp log(10^9) * B * G

Subtask 4:
Trường hợp k = 1, ta cần tìm 2 phần tử i và j sao cho |Bi - Gj| nhỏ nhất. Có nhiều cách làm, một trong số đó là:
- Sắp xếp mảng B và mảng G.
- Xét từng giá trị Bi, khi đó, chênh lệch |Bi - Gj| tạo thành hàm lồi

Cách làm này độ phức tạp BlogB + GlogG + BlogG

Một cách cải tiến bước 2 là rút ra nhận xét:
Giả sử, với i, ta tìm được vị trí j tốt nhất (gọi là j*). Bước tiếp theo với i+1, vị trí tốt nhất tìm được chắc chắn phải >= j* ==> ta có thể tìm j tuyến tính.

Subtask 6:
- Sắp xếp mảng B và mảng G.
- Chia nhị phân giá trị X là chênh lệch lớn nhất
    - Khi đó Bi có thể ghép với bất kì Gj nào trong đoạn [Bi - X, Bi + X]
    - Nhận thấy ta có B đoạn bằng nhau để ghép G giá trị vào và các đoạn này đã được sắp xếp tăng dần.
    - Tham lam, đi từ đoạn nhỏ đến đoạn lớn, nếu gặp giá trị j nào phù hợp thì ghép luôn.
- Độ phức tạp là BlogB + GlogG + log(10^9) * (B + G)

Bài 2. Kiểm định mã:
Subtask 1: Duyệt toàn bộ xâu nghiệm có thể và kiểm tra. Độ phức tạp là O(3^L*L^3) với L là độ dài xâu kết quả

Subtask 2: Giả sử A, B là hai dẫn xuất của xâu kết quả. Ta sẽ duyệt để xây dựng từng ký tự của A, B: Khi duyệt đến vị trí i, ta cần kiểm soát phần cuối của A đang là một hậu tố trong tập S.
Gọi x và y là hai tiền tố của tập S ứng với phần cuối của A, B. Khi đó (i, x, y) mô tả một trạng thái của quá trình duyệt. Gọi dp(i, x, y) là số ký tự ít nhất cần thêm vào A, B sao cho thu được hai dẫn xuất khác nhau của một xâu. Khi đó dp(i, x, y) = min(1+dp(i+1, x+c, y+c); c='a'..'z'); nếu x là một xâu trong S thì cần cập nhật thêm dp(i+1, '', y); nếu y là một xâu trong S thì cần cập nhật thêm dp(i+1, x, '')
Ở đây x, y có thể được thay thế bởi 1 node trên cây tiền tố (trie tree) của tập S. Độ phức tạp là O(L*sum^2) với L là độ dài xâu kết quả và sum là tổng độ dài các xâu trong S

Subtask 3: Dựng đồ thị có tập đỉnh là tập các cặp (x, y) với x, y là các tiền tố của tập S. Từ đỉnh (x, y) ta nối các cung có trọng số 1 đến (x+c, y+c) với c='a'..'z'; các cung có trọng số 0 đến ('', y) nếu x thuộc S; (x, '') nếu y thuộc S. Bài toán đưa về tìm đường đi ngắn nhất trên đồ thị này và có thể giải bằng thuật toán BFS
Độ phức tạp O(sum^2) với sum là tổng độ dài các xâu trong S

Bài 3.

Sub1: 1≤n, q≤2 000.
Duyệt hết tất cả các đỉnh trong tập để kiểm tra. Đpt O(N*Q).
Sub2: 1≤n, q≤50 000và mỗi đỉnh có tối đa 2 cạnh nối trực tiếp.
Cây trong trường hợp này có thể coi như 1 đường thẳng, các tập cần xét sẽ là 1 hoặc 2(trường hợp ngoài cây con) đoạn liên tiếp.
Ta có thể sử lý mỗi truy vấn theo thuật sau:

- Chọn ngẫu nhiên 30 phần tử trong tập.
- Với mỗi phần tử, kiểm tra xem màu của phần tử đó có thỏa mãn không, nếu có, thì đó là kết quả của truy vấn.
- Nếu xét hết 30 phần tử mà vẫn không tìm được màu thỏa mãn, ta chấp nhận in ra -1 với xác xuất sai rất nhỏ (1/2)^30.
- Để kiểm tra 1 màu có thỏa mãn không, ta có thể lưu mỗi màu 1 danh sách các vị trí xuất hiện và chặt nhị phân để đếm số lần xuất hiện trong đoạn.

Đpt : O(N + Q*30*log2(N)).
Sub3: 1≤n, q≤50 000và1≤ai≤2.
Ta lưu 2 mảng f[i][j] là số lần xuất hiện màu j trong cây con gốc i và g[i][j] là số lần xuất hiện màu j trên đường đi từ 1 đến I;
Với mỗi truy vấn, ta kiểm tra cả 2 màu, số lần xuất hiện của màu i:

- Truy vấn loại 1: f[u][i];
- Truy vấn loại 2:f[1][i]-f[u][i];
- Truy vấn loại 3:f[u][i]+f[v][i]-2*f[lca(u,v)][i]+a[lca(u,v)];

Dpt : O((N+Q)*log2(N)).

Sub4: 1≤n, q≤50 000và không có truy vấn loại3.
Ta dùng thuật toán dfs để duỗi cây ra, bây giờ các đỉnh thuộc cây con gốc u nằm liên tiếp nhau trong dãy khi duỗi.
Đến đây, do không có truy vấn loại 3 nên ta có thể xử lý như trong sub2.
Đpt : O(N + Q*30*log2(N)).
Sub4: không có ràng buộc nào thêm.
Đặt T=sqrt(N).
Với mỗi truy vấn ta chia làm 2 trường hợp:

- Nếu lực lượng của tập <= 2*T ta duyệt trâu để kiểm tra.
- Nếu lực lượng của tập > 2*T -> nếu tập đẹp thì tồn tại 1 màu xuất hiện >T lần. Đến đây ta có thể xử lý như sub3, nhưng thay vì xét 2 màu, ta xét hết các màu có số lần xuất hiện trên cây >T lần, có tối đa N/T màu như vậy.

Đpt : O(N*log2(N) + Q*sqrt(N)).