**Triangle**

* Sub1: n<=30: với mỗi đoạn (l, r) ktra tất cả các bộ ba phần tử thỏa mãn ko.
* Sub2: n<=1000: cố định r, lùi dần l khi mà tổng hai phần tử có giá trị nhỏ nhất lớn hơn phần tử có giá trị lớn nhất.
* Sub3: n<=1e5, a[i] = a[i-1]+1: có định r, lấy l xa nhất thỏa mãn a[l] + a[l+1] > a[r] ( giá trị của r càng lớn thì l càng lớn ).
* Sub4: n<=1e5: ý tưởng tương tự sub3 tuy nhiên vì ko phải dãy tăng dần nên ta lấy giá trị nhỏ nhất, nhì và lớn nhất bằng multiset ( có thể dùng IT hoặc RMQ)

**Triseq**

* Sub1: n<=9: sinh.
* Sub2: n<=18

+ f[id][mn][mn2][mx] là khi xây đến vị trí id và giá tị nhỏ nhất là mn, nhỏ nhì là mn2 và lớn nhất là mx thì có tất cả bao nhiêu cách xây tiếp.

+ với truy vấn 1 xuất f[1][n+1][n+1][0].

+ với truy vấn 2, 3 xem mảng f như một cấu trúc cây, các nhánh của cây đã được sắp xếp theo thứ tự từ điển để tìm kết quả ( tham khảo code )

**Lpass**

* Có 3 điều kiện của P:

+ P chia hết cho 9 -> tổng các chữ số của P chia hết cho 9 -> biến mod kiểm soát khi xây thì P chia 9 dư mod.

+ Chọn đúng k số -> dùng biến j để kiểm soát số số đã chọn.

+ P lớn nhất -> nhận thấy rằng với hai xâu a, b thì a+b != b+a nến ta sort mảng số yêu thích lại sao cho a[i] + a[i+1] > a[i+1] + a[i].

* Gọi f[i][j][mod] là số P tạo được khi xét đến số yêu thích thứ i, chịn j số, P đang chia 9 dư mod.

**Spath**

* Sub1: n<=700: tại mỗi đỉnh thì bfs để tìm đường đi ngắn nhất đến các đỉnh khác.
* Sub 2, 3: n<=1e4:

+ Với mỗi truy vấn i, j, swap(i, j) để bậc của i bé hơn.

+ Sort truy vấn theo i.

+ Với mỗi truy vấn nhóm i (các truy vấn có đỉnh bắt đầu là i), ta chia hai trường hợp:

\_ Bậc i >= n/2 thì ta sẽ thấy với mỗi j thuộc truy vấn nhóm i thì bậc của j cũng >=n/2. Khi đó đường đi từ i đến j có hai khả năng : đi trực tiếp từ I đến j hoặc đi từ j đến u rồi đến j ( u bất kì).

\_ Bậc i < n/2, ta loang từ đỉnh i để đi đến các đỉnh của truy vấn nhóm i.