**Sol\_network**

Trước tiên, ta có nhận xét, với bộ 4 đỉnh cho ta 2 trường hợp:

+ Tứ giác lồi, khi đó sẽ có 2 hình tròn tạo bởi 3 điểm chứa đỉnh còn lại, gọi số tứ giác lồi là x

+ Tứ giác lõm, có 1 hình tròn tạo bởi 3 điểm chứa đỉnh còn lại, gọi số tứ giác lõm là y Vậy kết quả bài toán là (2x + y + 3 \* nC3) / nC3

Để tính được x và y, ta sẽ sử dụng các phương trình trung gian: x + y = nC4

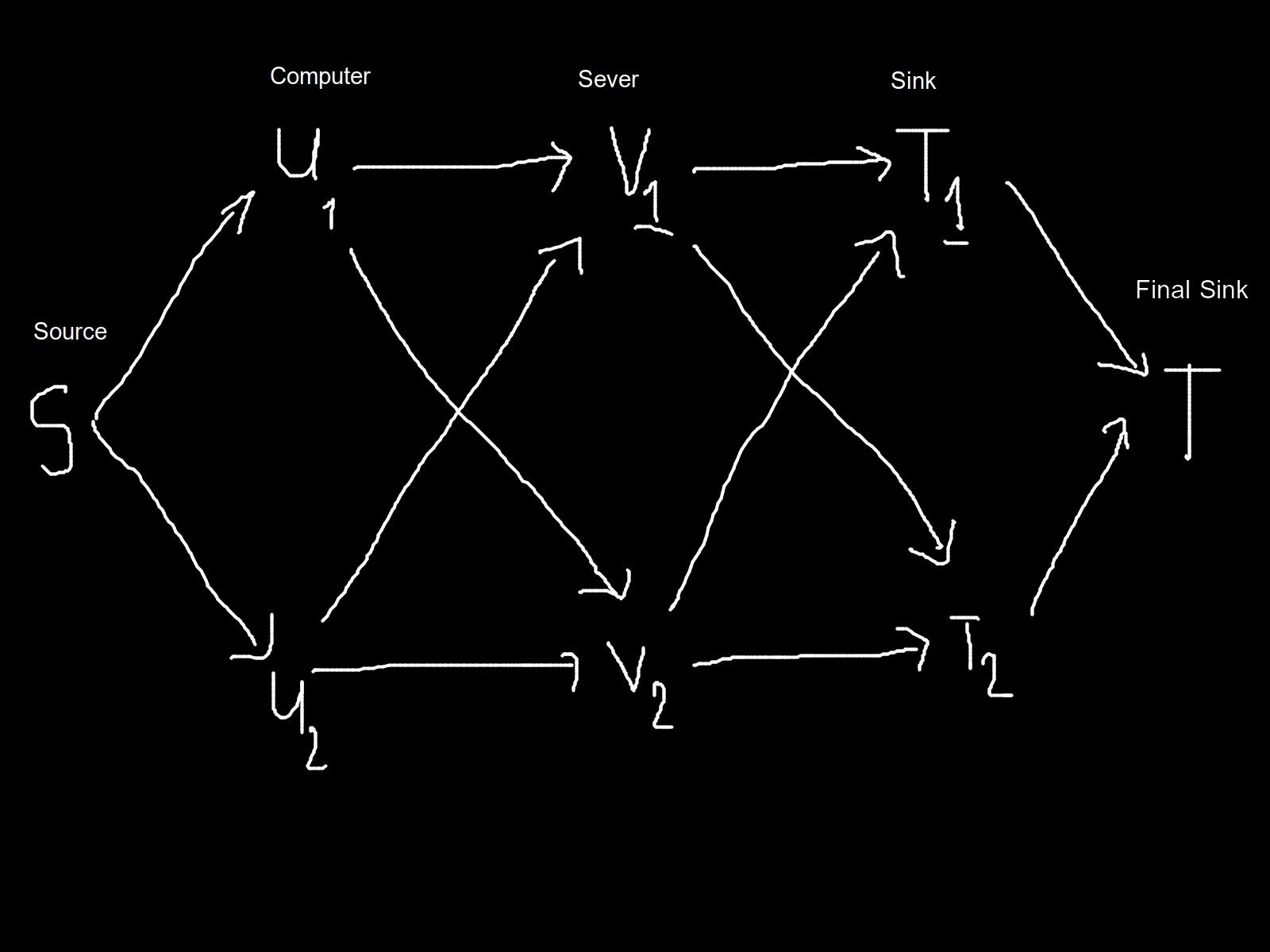
Ngoài ra, với mỗi hình tứ giác lồi, sẽ có 4 tia nối bởi 2 đỉnh nằm trong 2 cạnh hình còn tứ giác lõm thì là 3

4x + 3y = S trong đó S là số tia nằm giữa 2 tia.

Đến đây thì đơn giản hơn rồi, ta sẽ cố định O là gốc của tia, sort các đỉnh còn lại theo ccw, với mỗi điểm A ta sẽ trượt tìm điểm B xa nhất mà góc A0B < 180

Lưu ý, cách này làm được bởi không có 3 điểm nào thẳng hàng và tứ giác nội tiếp.

**Sol\_Wifi**

Bài toán được chuyển về luồng như sau:

Trong đó 2 đỉnh S, T là source và sink của luồng, ui và vi là đỉnh đại diện cho máy tính và sever nhận dữ liệu, đỉnh T1, T2 là các đỉnh phụ ta thêm vào giúp giải quyết bài toán.

Capacity và Cost của các cạnh lần lượt là:

(S, Ui) = 1, 0

(Ui, Vi) = 1, distance (Ui, Vi) (Vi, T1) = limit, 0

(Vi, T2) = 1, 0

(T1, T) = limit \* m, 0 (T2, T) = rem, 0

Với limit = n div m, rem = n mod m. Giải thích: Với mỗi máy, ta có cạnh

+ S -> Ui mang ý nghĩa sẽ dùng 1 máy đó để kết nối

+ Ui -> Vj tượng trưng cho việc máy i sẽ liên kết với sever j

+ Vj -> T1 = limit vì mỗi máy bắt buộc phải vận chuyển một lượng ít nhất bằng limit, rõ ràng nếu mang ít hơn thì sẽ có máy mang limit + 1 và khi đó chênh lệch sẽ lớn hơn 1.

+ Vj -> T2 = 1 vì sẽ có thừa ra một vài máy tính và những cạnh nối này có nhiệm vụ xử lý nốt

Đến đây, bài toán là Min Cost Max Flow.

**Sol\_Supervisor**

Bài toán cho ta một rừng các cây.

+ Đối với cây đã là perfect matching thì ta chỉ ghép nó vào 1 vì không tối ưu được nữa.

+ Đối với cây chưa phải thì ta xét 2 trường hợp, nếu sử dụng và không sử dụng gốc thì sẽ được bao nhiêu, từ đó mình ghép tham vào một đỉnh free để tăng kết quả lên 1