Анализа проблема

Праволинијско решење се добија тако што се обрађује свака четворка тачака и утврђује да ли је површина четвороугла који оне одређују ближа задатој површини. При одређивању површине четвороугла које одређују тачке A, B, C и D треба искористити чињеницу да бар једна од његових дијагонала припада четвороуглу. Та дијагонала дели четвороугао на два троугла, а површина четвороугла ће бити једнака збиру површина та два троугла.

Како одредити ту дијагоналу? Па ако нпр. дуж AC представља ту дијагоналу онда су тачке B и D са различитих страна праве коју одређују тачке A и C. Проверу можемо извести тако што ћемо кроз тачке A и C поставити праву. Ако су x_A и y_A координата тачке A, а x_C и y_C координате тачке C, онда је једначина праве

$$\frac{x - x_A}{x_C - x_A} = \frac{y - y_A}{y_C - y_A}$$

или

$$(x - x_A)(y_C - y_A) - (y - y_A)(x_C - x_A) = 0$$

Ако нека трећа тачка не припада правој постављеној кроз тачке A и C, онда израз који добијамо када у левој страни горњег израза заменимо x и y са координатама те тачке има вредност различиту од нуле. Ако су вредности које се добију када се x и y замене координатама тачке B, односно координатама тачке D различитог знака, онда су тачке B и D са различитих страна праве AC.

Како одредити површину троугла чија су темена тачке $A(x_A, y_A)$, $B(x_B, y_B)$ и $C(x_C, y_C)$? Можемо формирати векторе $\vec{AB} = (x_B - x_A, y_B - y_A)$ и $\vec{AC} = (x_C - x_A, y_C - y_A)$. Онда векторски производ ова два вектора представља вектор чија је дужина једнака површине паралелограма разапетог над тим вектроима. Аритметички, израз

$$\begin{vmatrix} x_B - x_A & y_B - y_A \\ x_C - x_A & y_C - y_A \end{vmatrix} = (x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A)$$

може бити и позитиван и негативан (али исто тако и нула, ако су тачке A, B и C колинеарне), а његова апсолутна вредност је једнака баш површини паралелограма разапетог над векторима $\vec{AB}$ и $\vec{AC}$, односно двострукој површини троугла ABC. Напоменимо да ако си изрази

$$(x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A) \qquad (x_B - x_A)(y_D - y_A) - (x_D - x_A)(y_B - y_A)$$

(су координате тачке), различитог знака, онда су тачке C и D са различитих страна праве AB.

Напреднији алгоритам добијамо тако што анализирамо све могуће парове тачака као кандидате за дијагоналу четвороугла и то ону дијагоналу која припада унутрашњости четвороугла. Ако су A и B кандидати за крајеве дијагонале, онда остале тачке делимо у две групе, зависно од тога са које стране праве AB се налазе. Нека су $P'_1, P'_2, \ldots, P'_{n'}$ површине троуглова који се налазе са једне стране праве, а $P''_1, P''_2, \ldots, P''_{n''}$ површине троуглова који се налазе са друге стране праве AB. Та два низа сортирамо (нпр. у неопадајућем поретку). Након тога се симултано крећемо кроз низове покушавајући да суму површина троуглова из та два низа максимално приближимо траженој вредности. Због тога по једном од низова крећемо од почетка (тј. од најмање површине), а по другом од краја (тј. од највеће површине). Ако је тренутни збир мањи од циљане вредности, померамо се по низу по коме идемо од почетка (и на тај начин повећавамо збир). Ако је површина већа од циљане вредности, померамо се по низу по коме се крећемо од краја. Наравно, прекидамо у тренутку када потрошимо један од низова. Сложеност дела који се односи на обраду једног пара тачака (као кандидата за крајеве дијагонале) је $\Theta(n \log n)$ (сортирање низова са површинама). Како је број парова тачака, то је сложеност комплетног поступка $\Theta(n^3 \log n)$.

Алгоритамске смернице

Прилажемо само блок програма у коме се обрађује сваки пар тачака као крајеви дијагонале. Препуштамо читаоцу да допуни са блоком за учитавање и испис резултата.

```
for (int i=0; i<n-1; i++) {
    for (int j=i+1; j<n; j++) {
        left_cnt = 0;
        right_cnt = 0;
        for (int k=0; k<n; k++) {
            if (k == i || k == j) continue;
            Triangle t;
            t.index = k;
            t.area = area(i, j, k);
            if (t.area > 0)
                tleft[left_cnt++] = t;
            else if (t.area < 0) {
                t.area = -t.area;
                tright[right_cnt++] = t;
            }
        }
        if (left_cnt == 0 || right_cnt == 0) continue;
        sort(tleft, tleft + left_cnt);
        sort(tright, tright + right_cnt);
        //We move from opposite sides of sorted arrays,
        // and find the closest area
        pos_left = 0;
        pos_right = right_cnt-1;
        while (pos_left < left_cnt && pos_right >= 0) {
            sum = tleft[pos_left].area + tright[pos_right].area;
            difference = abs_val(sum - 2 * target_area);
            if (difference < min_difference ||
               (difference == min_difference && sum > sol)) {
                min_difference = difference;
                sol = sum;
            }
            if (sum > 2 * target_area)
                pos_right--;
            else
                pos_left++;
        }
    }
```