Z1

Algorytm:

```
def gcd(a, b):
    if b > a:
        return gcd(b, a)
    if b == 0:
        return a
    if b == 1:
        return 1
    if a % 2 == 0 and b % 2 == 0:
        return 2 * gcd(a/2, b/2)
    if a % 2 == 0 and b % 2 == 1:
        return gcd(a/2, b)
    if a % 2 == 1 and b % 2 == 0:
        return gcd(a, b/2)
    return gcd(abs(a-b)/2, b)
```

Poprawność wynika wprost z poprawności własności z zadania.

Przechodzimy złożoności:

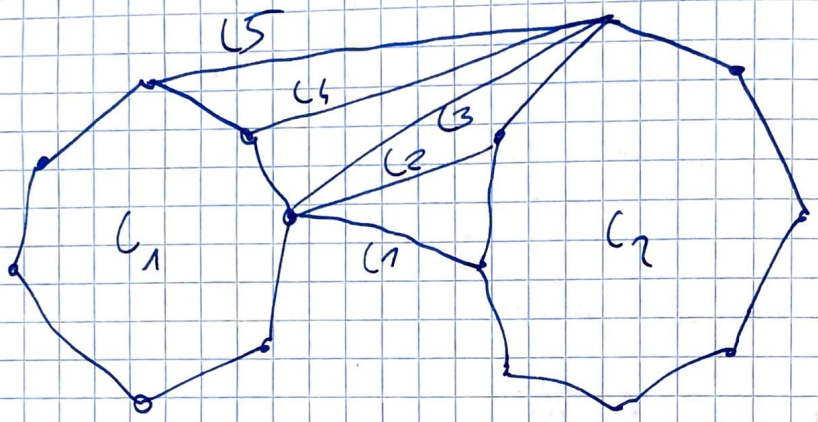
- Zauważmy, że w max dwóch kolejnych wywołaniach rekurencyjnych jeden z argumentów maleje dwukrotnie. Zatem w najgorszym wypadku wykonamy $T(a,b) = 2(\log a + \log b)$ operacji, czyli $O(\log ab)$, tak jak Euklides.

Z3|

Mamy dane dwa zbiory wierzchołków C_1 i C_2, które są otoczkami wypukłymi.

Idea: znajdziemy takie styczne, które potrafią ominąć otoczki w jedną.

Ilustracja:



Algorytm znajdowania styczych:

while (L przecina C_1 lub C_2):
    while (L przecina C_i):
        $L \leftarrow L'$ (punkt na C_2 idzie w górę)
    while (L przecina C_1):
        $L \leftarrow L'$ (punkt na C_1 idzie w górę)

Dla dolnej analogicznie z przeciwnym w dół.

Złożoność: $O(n)$

Poprawność:

- Otrzymujemy wielokąt wypukły, wynika to z nieprzecinania się stycznych z C_1 i C_2
- Ten wielokąt zawiera wszystkie punkty ze zbioru $S_1 \cup S_2$
- Wynika wprost z działania algorytmu oraz z faktu, że S_1 i S_2 są otoczkami

(Obserwacja: każdy punkt zbioru $S_1 \cup S_2$ należy do S_1 lub S_2)

- wielokąt O_1 jest wypukły.

Załóżmy nie wprost, że nie jest. Czyli istnieje inny wielokąt O_2 wypukły zawierający wszystkie punkty z $C_1 \cup C_2$, tzn że istnieje jakiś punkt należący do O_2 i nie należący do O_1. Ale to oznaczałoby utratę wypukłości. Sprzeczność.

Przykład: