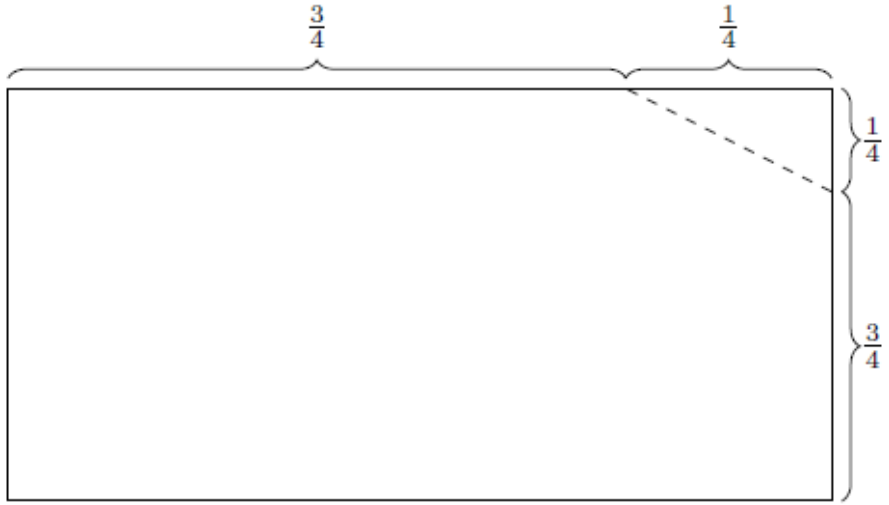
Problem E

A problem of packaging

Marco Lauer, Luca Lauer

Aufgabenstellung

- Sophie hat einen Kuchen mit konvexer Form für die Hochzeit ihrer Freundin gebacken
- Leider ist der Kuchen zu schwer für den Transportservice
- Sophie muss Teile abschneiden, um den Kuchen leichter zu machen
- Sie muss eine Zahl s ($2 \leq s \leq 1000$) wählen, die die Größe der abzuschneidenden Ecken darstellt
- Für jede Ecke markiert sie 2 Stellen, die jeweils $Länge\ der\ Kante/s$ auf den inzidenten Kanten entfernt sind
- Diese markieren die Gerade für den Schnitt
- Sie möchte s so wählen, dass möglichst viel vom Kuchen übrig bleibt, er aber leicht genug für den Transportservice ist



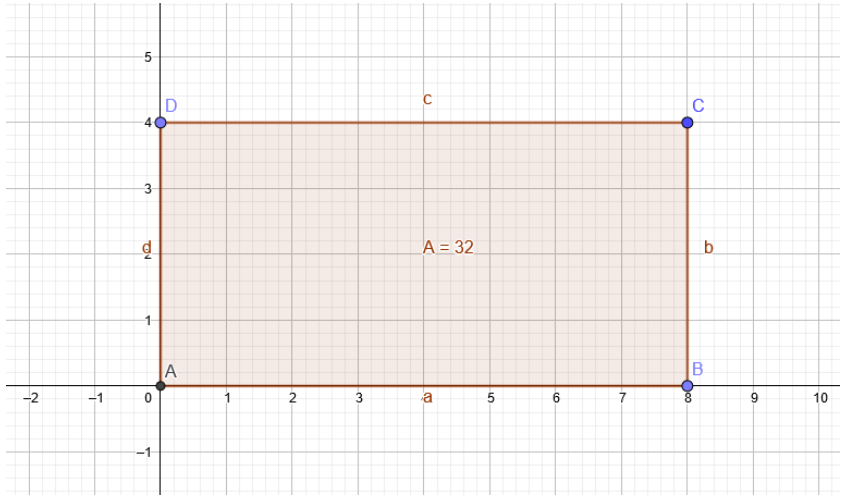
Einer der vier Schnitte für $s = 4$ bei einem Rechteck

Input

1. Zeile:
   - Float a: Verhältnis, auf welches Gewicht des Kuchens mindestens verringert werden muss ($0.25 \leq a < 1$)
   - Integer N: Anzahl der Ecken des Kuchen ($3 \leq N \leq 100$)

N Zeilen:
   - Integer x_i, y_i: Koordinaten der jeweiligen Ecke ($0 \leq x_i, y_i \leq 10^8$)
   - Die Ecken werden in der Reihenfolge gegeben, in der sie eine konvexe Form bilden

```
0.875 4
0 0
8 0
8 4
0 4
```



Output

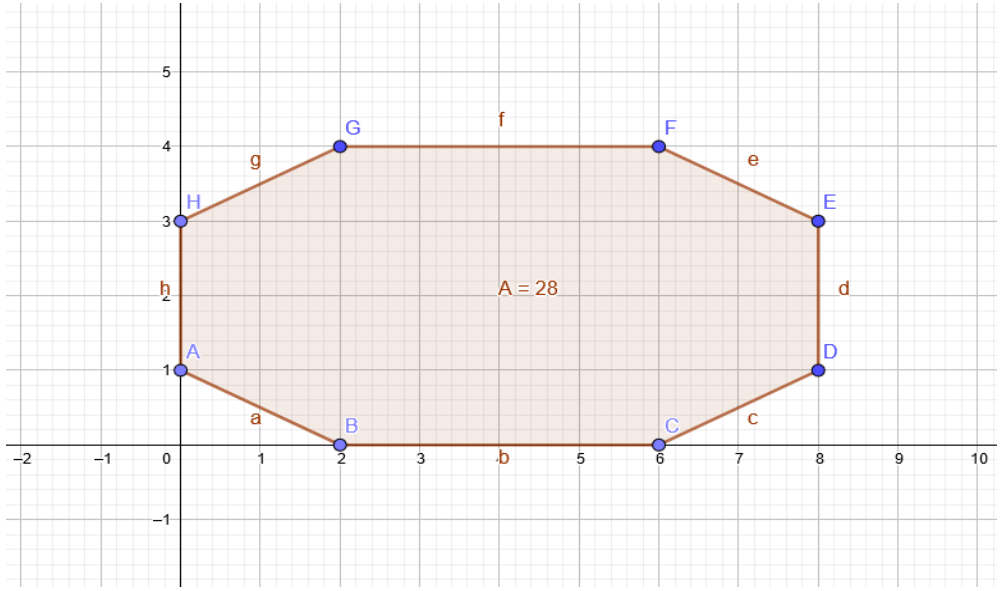
- Größtmögliche s ($2 \leq s \leq 1000$), sodass der Flächeninhalt höchstens $a *$ *ursprünglichen Flächeninhalt* ist

Erlaubter absoluter Fehler: 10^{-4}, d.h. $|berechneter\ Wert - gesuchter\ Wert| \leq 10^{-4}$

Ursprünglicher Flächeninhalt: 32

Gesuchter Flächeninhalt: $32 * 0.875 = 28$

Diesen erreichen wir bei $s = 4$



Herausforderungen

- Berechnung des Flächeninhalts eines konvexen Polygons
- Finden des besten s

Berechnung des Flächeninhalts

Shoelace Formula:

Berechnet Flächeninhalt eines einfachen Polygons:

$$A = \frac{1}{2}\left|\sum_{i=1}^{n}(x_i y_{i+1} - y_i x_{i+1})\right|$$

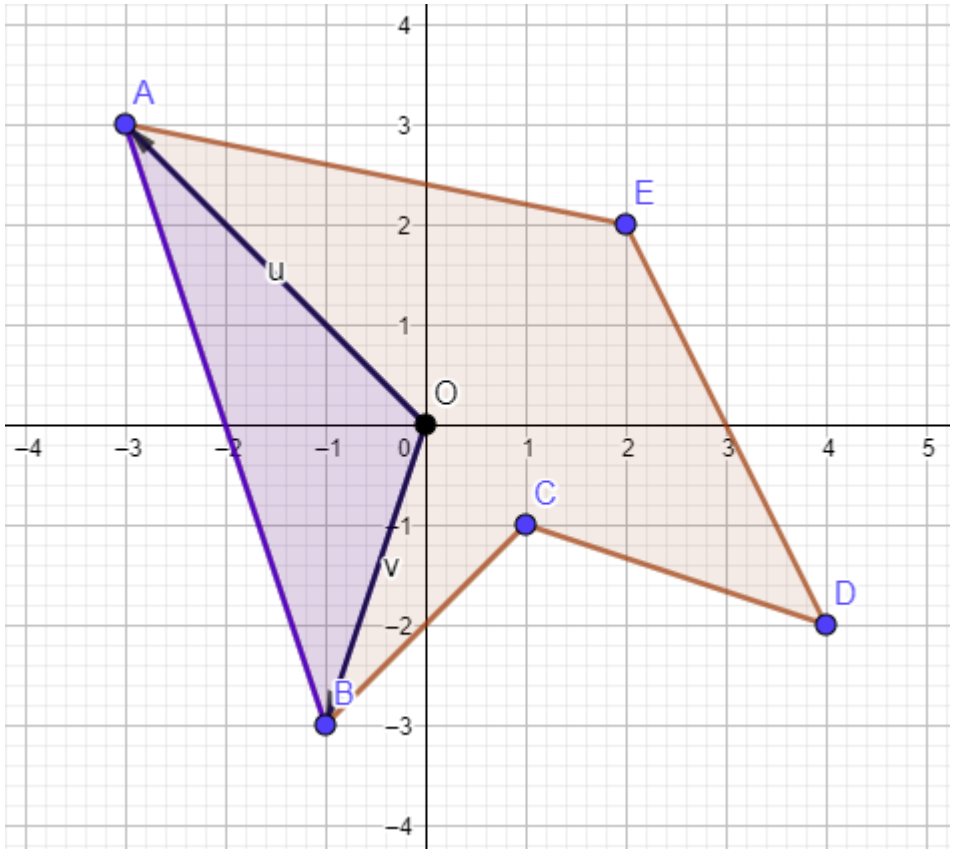
Wobei:

- n: Anzahl der Knoten
- (x_i, y_i): Koordinaten des Knoten i
- $x_{n+1} = x_1$ und $y_{n+1} = y_1$

Funktionsweise

Knoten: $(-3, 3), (-1, -3), (1, -1), (4, -2), (2, 2)$

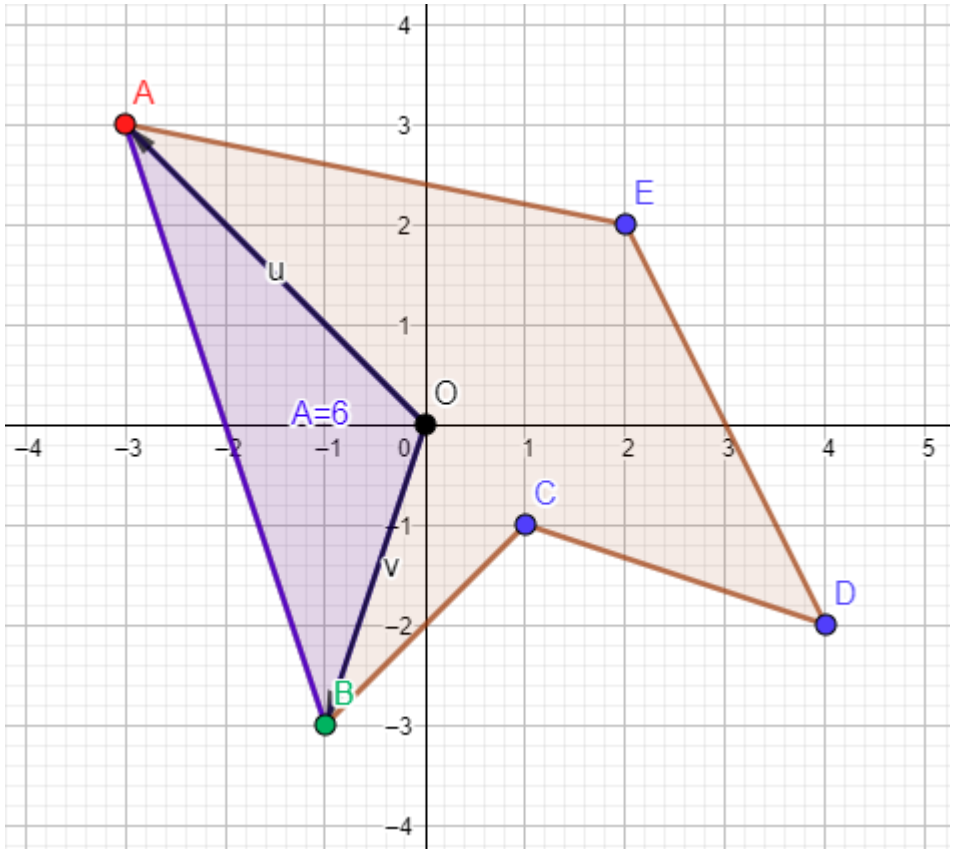
In $\mathbb{R}^2$: $A_{\blacktriangle} = \frac{1}{2}|det(\vec{u}\vec{v})| = \frac{1}{2}|(u_x v_y - u_y v_x)|$



Funktionsweise

Knoten: $(-3, 3), (-1, -3), (1, -1), (4, -2), (2, 2)$

In $\mathbb{R}^2$: $A_{\blacktriangle} = \frac{1}{2}|det(\vec{u}\vec{v})| = \frac{1}{2}|(u_x v_y - u_y v_x)|$



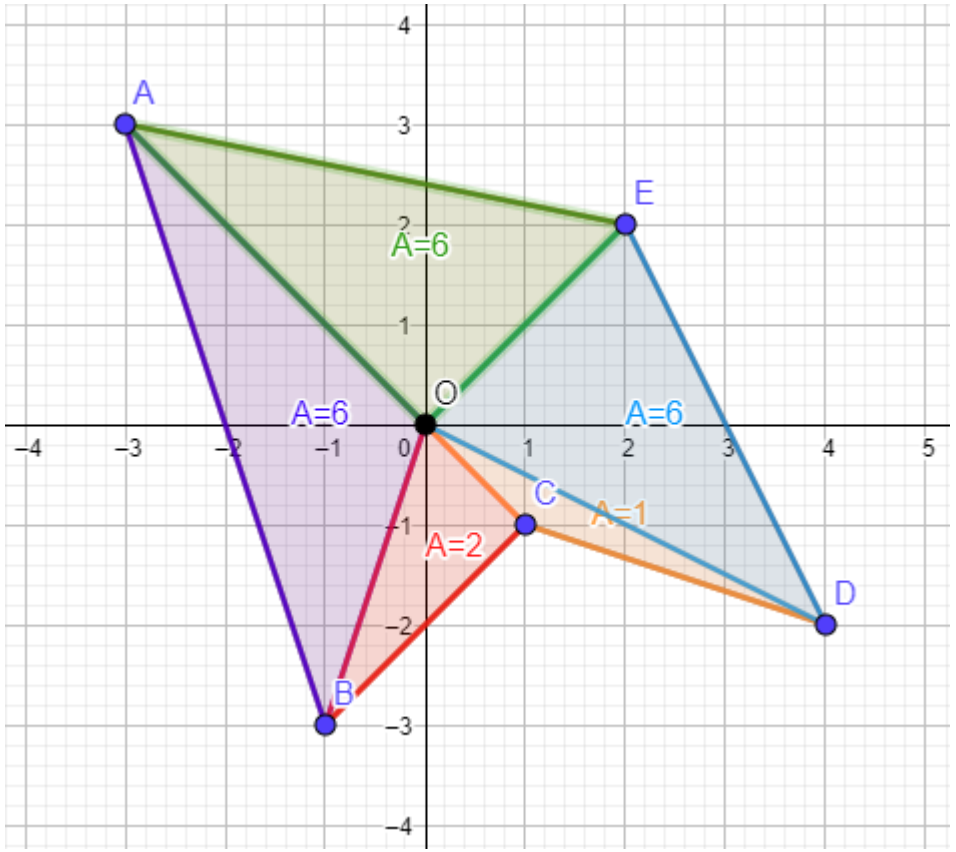
$$A = \frac{1}{2}\left|\begin{vmatrix} -3 & -1 \\ 3 & -3 \end{vmatrix}\right| = \frac{1}{2} * |(-3 * (-3) - (-1) * 3)| = 6$$

Funktionsweise

Knoten: $(-3, 3), (-1, -3), (1, -1), (4, -2), (2, 2)$

In $\mathbb{R}^2$: $A_{\blacktriangle} = \frac{1}{2}|det(\vec{u}\vec{v})| = \frac{1}{2}|(u_x v_y - u_y v_x)|$

$$\Rightarrow A = \frac{1}{2}\left|\sum_{i=1}^{n}(x_i y_{i+1} - y_i x_{i+1})\right|$$



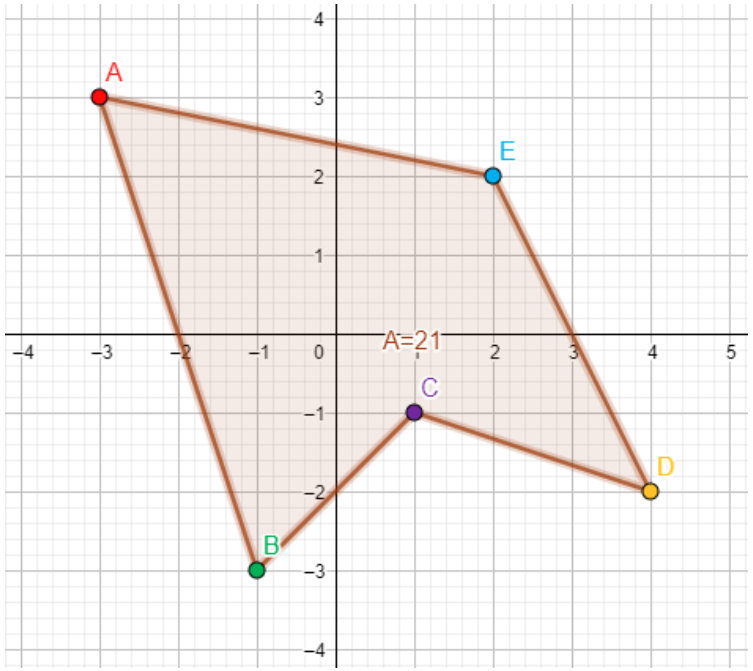
Berechnung des Flächeninhalts (Beispiel)

Knoten: $(-3, 3)$, $(-1, -3)$, $(1, -1)$, $(4, -2)$, $(2, 2)$

$$A = \frac{1}{2}\left|\sum_{i=1}^{n}(x_i y_{i+1} - y_i x_{i+1})\right|$$

$$A = \frac{1}{2}\left|\begin{array}{r} -3 * (-3) - 3 * (-1) + (-1) * (-1) - (-3) * 1 + 1 * (-2) \\ -(-1) * 4 + 4 * 2 - (-2) * 2 + 2 * 3 - 2 * (-3) \end{array}\right|$$

$$A = \frac{1}{2}|42| = 21$$



Finden des besten s

- Ausprobieren für alle Möglichkeiten von s ($2 \leq s \leq 1000$) mit Fehlertoleranz 10^{-4}

➔ Zu viele Berechnungen

- Mit steigendem s steigt die übrige Fläche monoton an

➔ Binärsuche

Binärsuche

```python
def binary_search(low, high, a, old_area):
    best = -1                       # Beliebige Initialisierung

    while (high-low) > 1e-6:        # Bei Abbruch könnte sich s nur noch um maximal 1e-6 verändern
        s = (high + low) / 2

        if new_area(s) > a*old_area:
            high = s
        else:
            best = s                # Speichere bestes Zwischenergebnis, das die Vorraussetzung noch erfüllt
            low = s

    return best
```

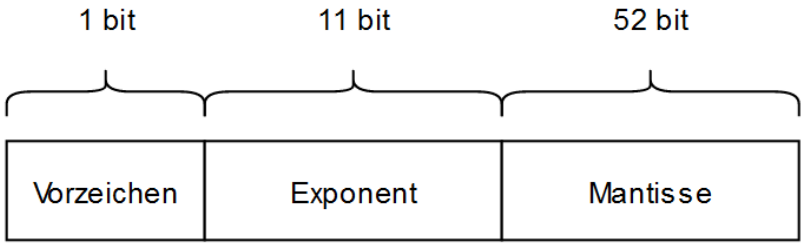
Laufzeit:

- $\lceil \log_2((high - low) * 10^{Anzahl\ Nachkommastellen}) \rceil$
- Mit 6 Nachkommastellen: $\lceil \log_2(998 \cdot 10^6) \rceil = 30$ Durchläufe der while-Schleife

Weitere Herausforderung: Größe der Mantisse von double

- $2^{52} \approx 4{,}5 \cdot 10^{15}$
- Koordinate kann vor dem Komma schon 10^8 groß sein
- In der Berechnung des Flächeninhalts müssen möglicherweise zwei solcher Zahlen multipliziert werden
- $10^{16} > 4{,}5 \cdot 10^{15}$

➔ Größe der Mantisse reicht nicht aus



$$\underbrace{4{,}5}_{\text{Mantisse}} \cdot \underbrace{10^{15}}_{\text{Exponent}}$$

Alternativen zum double

- Java

```java
public class BigDecimalTest {
    public static MathContext context = new MathContext( setPrecision: 50);     //Mantisse von 50 Stellen
    public static void main(String[] args) {
        BigDecimal sum = BigDecimal.valueOf(2).add(BigDecimal.valueOf(5));  // 2+5
        BigDecimal division = BigDecimal.valueOf(1).divide(BigDecimal.valueOf(7), context); // 1/7
    }
}
```

- Python

```
>>> from decimal import *
>>> context = Context(prec=50)
>>> setcontext(context)
>>> Decimal(1) / Decimal(7)
Decimal('0.14285714285714285714285714285714285714285714285714')
```

Laufzeit

- Binärsuche: Immer $\lceil \log_2(998 \cdot 10^6) \rceil = 30$ Durchläufe
- Shoelace-formula: $\Theta(n)$ (n: Anzahl der Ecken)

$\Rightarrow \Theta(n)$

- Jedoch: kleiner Mehraufwand, da wir nicht einfach mit double rechnen können
- Die Laufzeitkomplexität ändert sich aber nicht, weil die Koordinaten in ihrer Größe begrenzt sind

Viel Erfolg beim Implementieren :-)