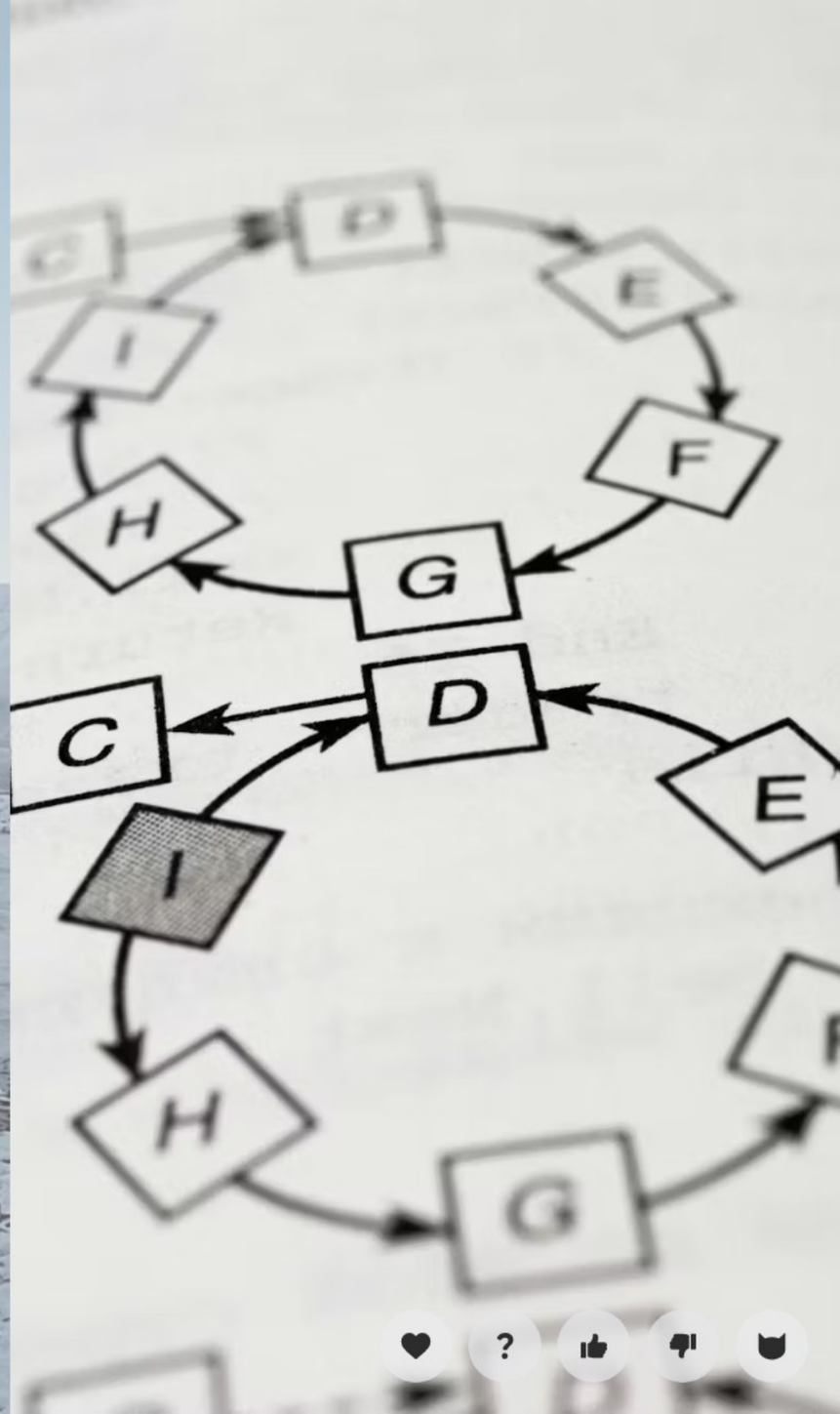
Algorytmy i Struktury Danych

26.10.2022 12:49



Rekurencja

- *recurrere* - przybiec z powrotem
- również rekursja z angielskiegu *recursion*
- Wywołanie funkcji w jej własnym ciele
- f(args) -> ... f(args') ... return

Przykład – silnia

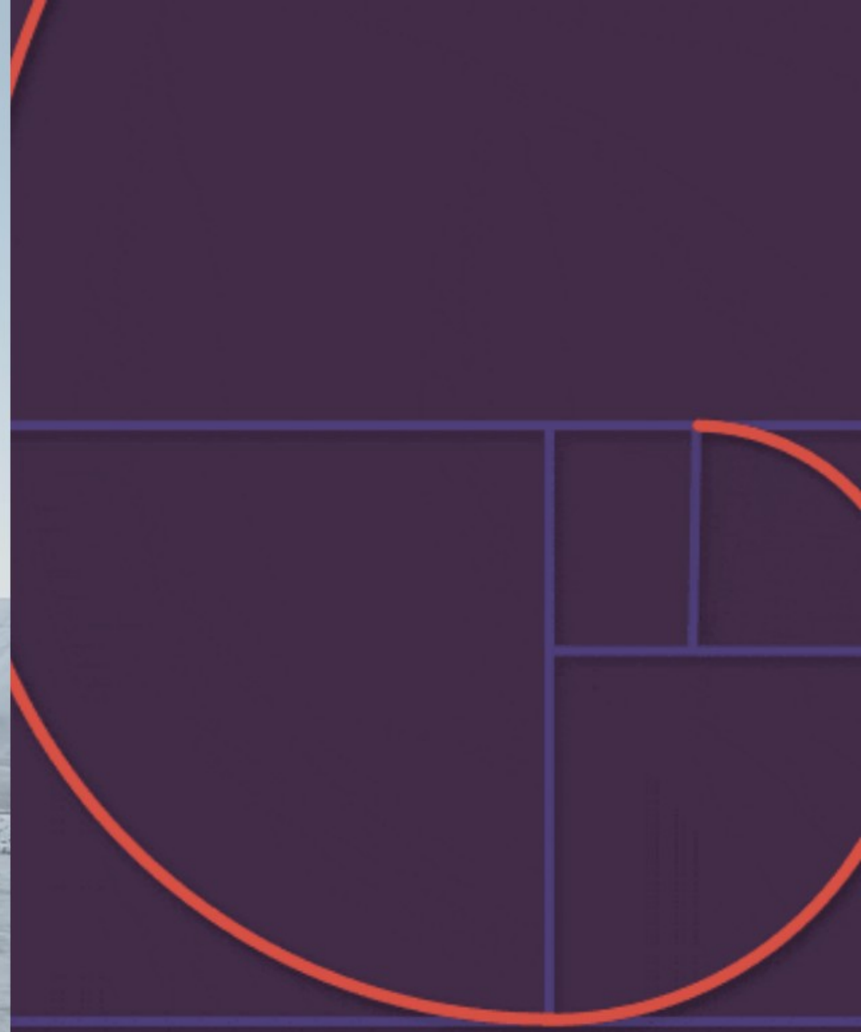
- $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5 \cdot 4!$
- $n! = n \cdot (n - 1)!$
- $0! = 1$

Przykład - silnia

- n! = n • (n - 1)!, 0! = 1
- Jak to zaimplementować w Pythonie?
- (-1)! = ?
- 0.5! = ?

Przykład – liczby Fibonacciego

- F(n) = F(n – 1) + F(n – 2)
- F(0) = 0, F(1) = 1
- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55...



Przykład - liczby Fibonacciego

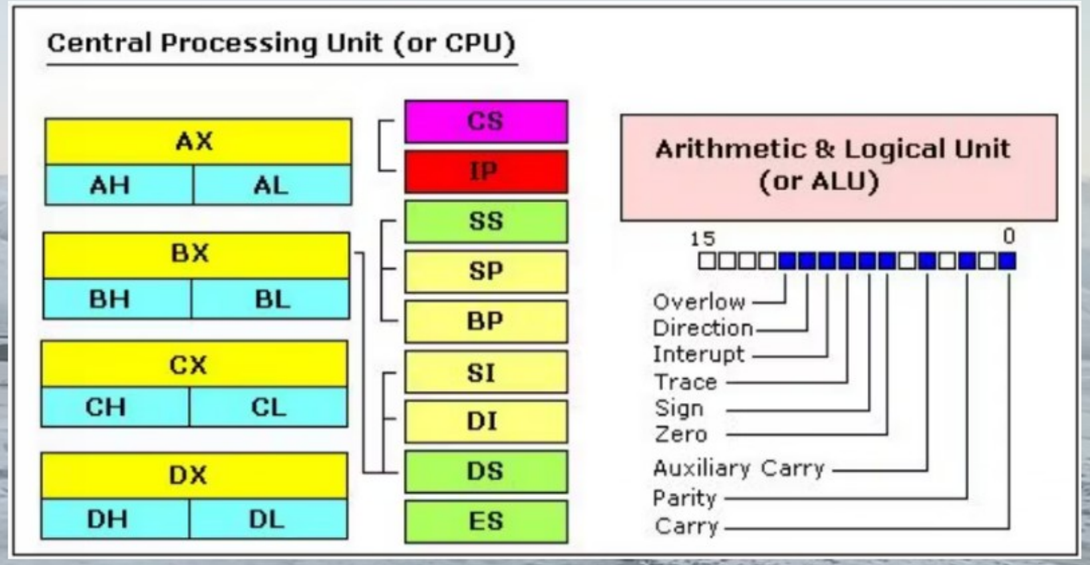
- F(n)= F(n - 1) + F(n - 2), F(0)= 0, F(1)= 1
- Jak to zaimplementować w Pythonie?
- F(-1) = ?
- F(0.5) = ?
- F(10)=?,F(20)=?,F(30)=?... F(100)=?
- Ile wywołań rekurencji będziemy mieli dla n-tej liczby Fibonacciego?
- ~ϕ^n, jakie są to wartości dla różnych n?
- Ile czasu zajmie wykonanie operacji dla n=100

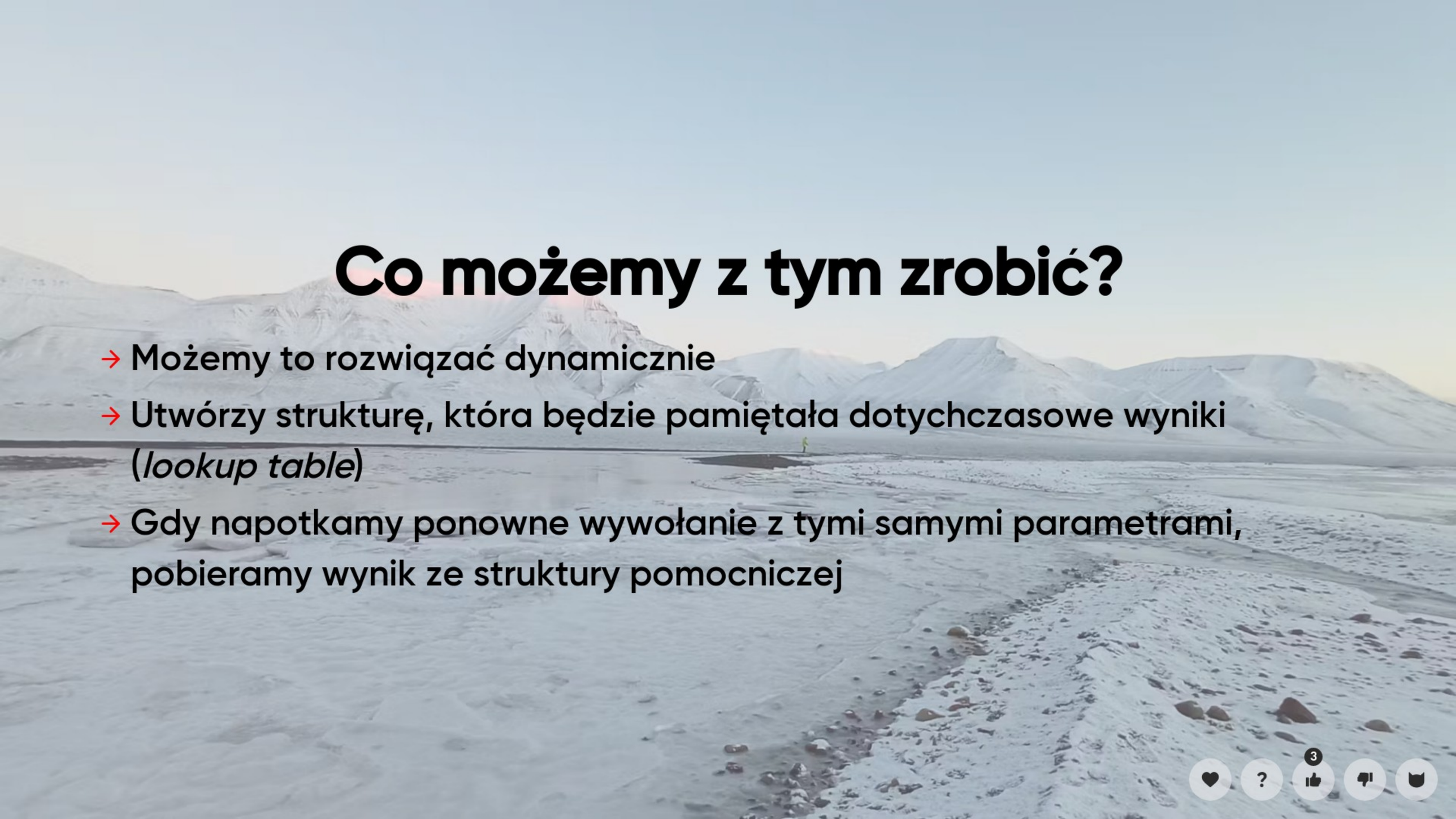
Głębokość rekurencji

- W pakiecie *sys* istnieje funkcja *setrecursionlimit*
- Jeśli chcemy, możemy ustawić ją nawet na miliardy
- Czemu to nie jest dobry pomysł?

Każdy proces ma swój kontekst, rejestry, stertę i stos

- kontekst - zestaw zmiennych wykorzystywanych przez aktualnie wywoływaną funkcję.
- Rejestry - zestaw komórek pamięci podręcznej dostępnej dla procesora natychmiastowo.
- Stos - obszar pamięci przechowujący konteksty odłożonych tymczasowo wywołań funkcji.
- Polecam poczytać o **syscall**





Co możemy z tym zrobić?

- Możemy to rozwiązać dynamicznie
- Utwórzy strukturę, która będzie pamiętała dotychczasowe wyniki (*lookup table*)
- Gdy napotkamy ponowne wywołanie z tymi samymi parametrami, pobieramy wynik ze struktury pomocniczej

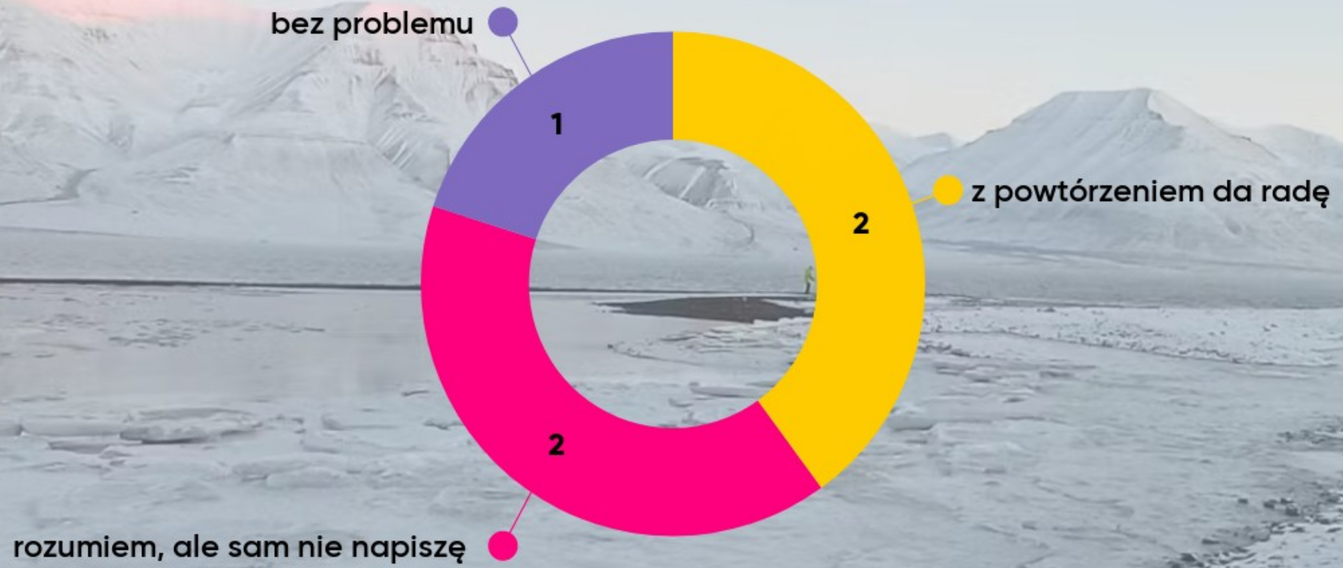
Co możemy z tym zrobić?

- Każdą rekurencję można zmienić w pętlę
- Proste dla silni i liczb Fibonacciego
- Potrafi być tragicznie złożone dla bardziej zaawansowanych wzorów
- Spróbujcie napisać silnię i liczby Fibonacciego w postaci pętli
- Liczby Fibonacciego wymagają n iteracji pętli zamiast wcześniejszych ϕ^n wywołań

Czy w ogóle potrzebujemy rekurencji?

- Rekurencja jest czytelniejsza
- Szybsza w zapisie
- Niektóre języki uwzględniają rekurencję ogonową, która jest faktycznie pętlą o wyglądzie rekurencji

Jak się czujecie z tym tematem?



5

Złożoność