Technologie cyfrowe

Artur Kalinowski

Zakład Cząstek i Oddziaływań Fundamentalnych

Pasteura 5, pokój 4.15

Artur.Kalinowski@fuw.edu.pl

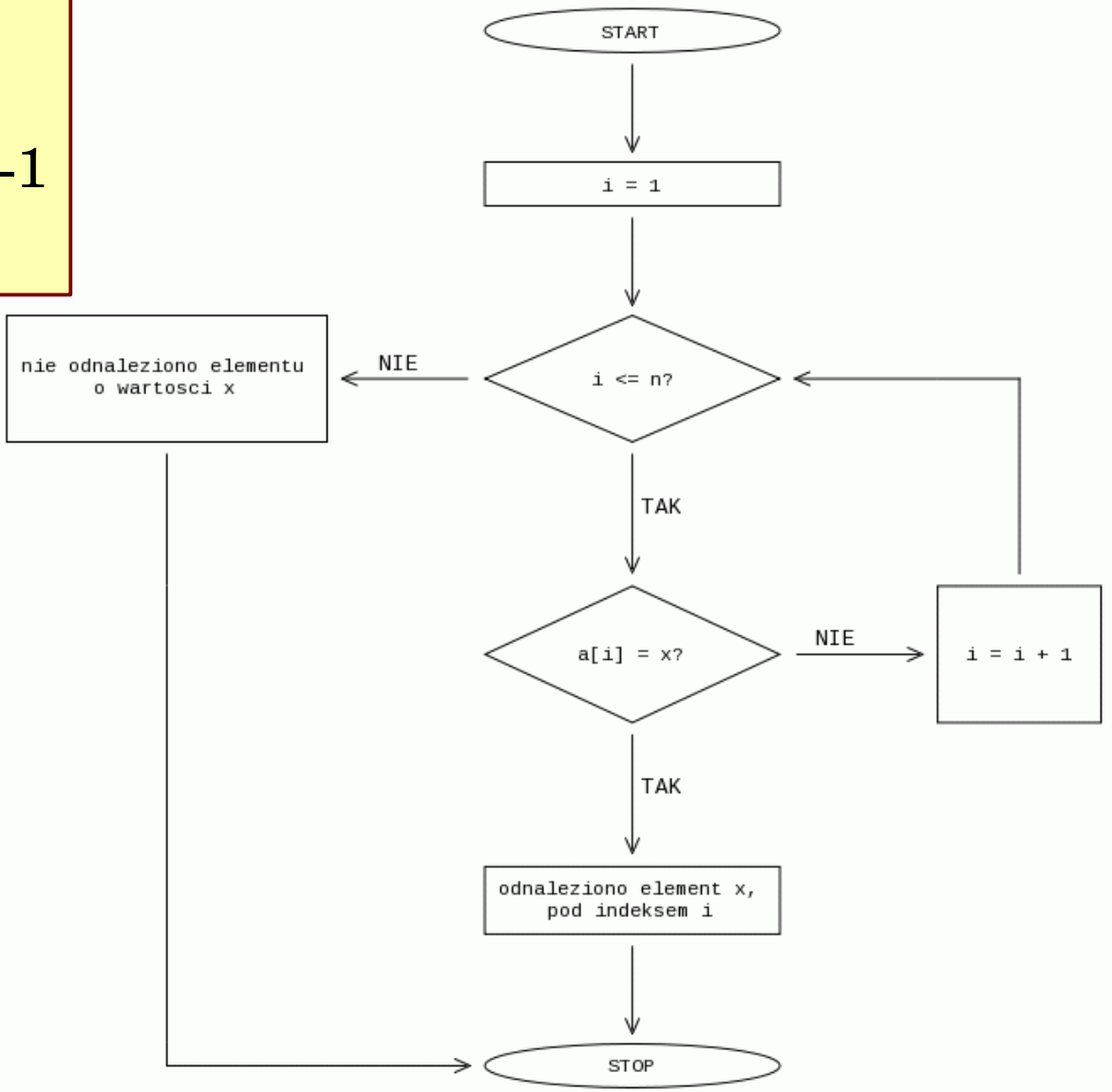
Semestr letni 2014/2015

Zadanie algorytmiczne: wyszukiwanie

dane wejściowe: lista elementów L, szukany element x
wynik: indeks elementu x, lub -1 jeśli elementu nie ma na liście

Przeszukiwanie liniowe: porównuj kolejne elementy z szukanym elementem x.

Złożoność obliczeniowa: O(n)



http://www.algorytm.org/dla-poczatkujacych/szukanie-zadanego-elementu.html

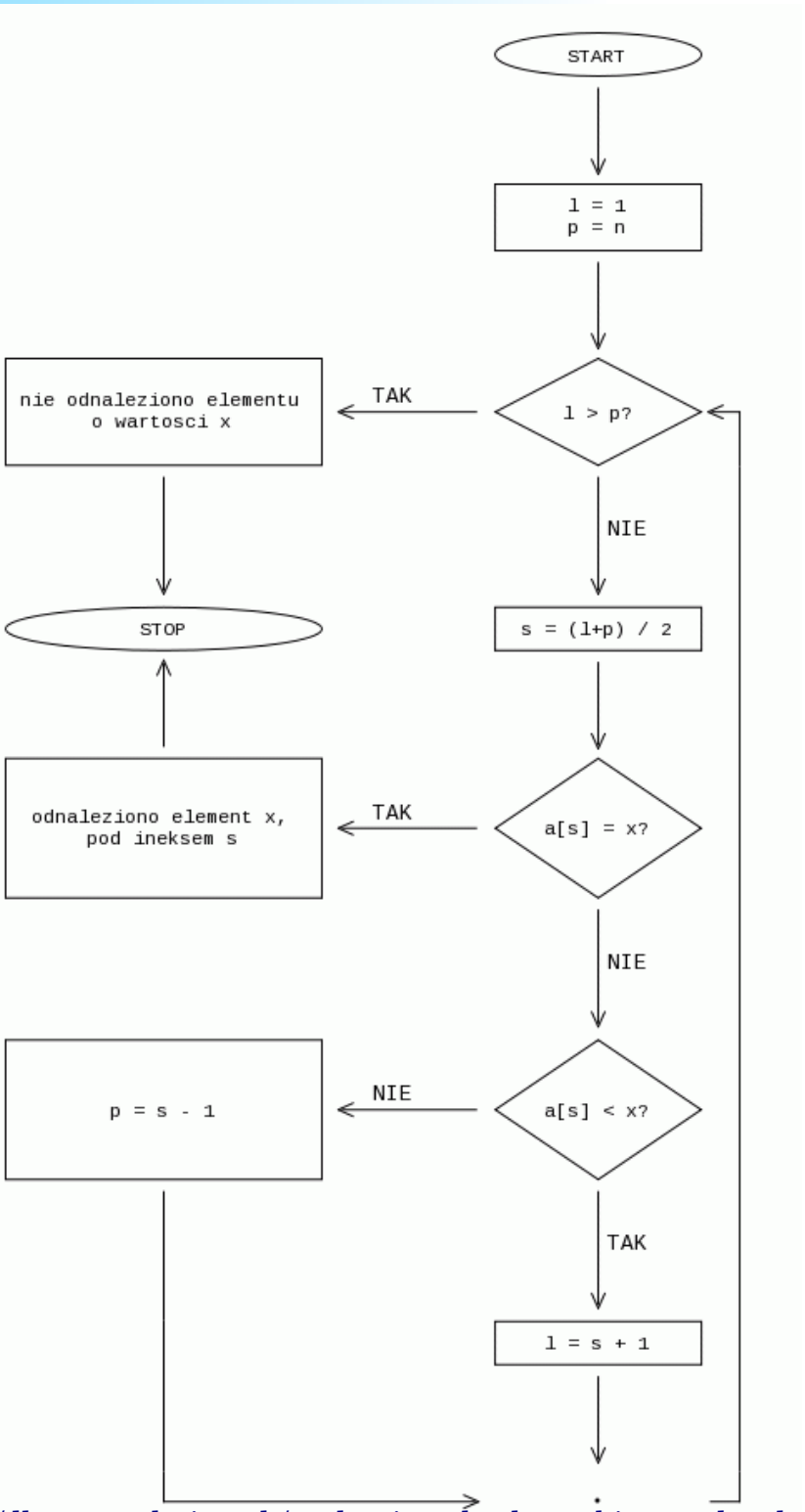
Zadanie algorytmiczne: wyszukiwanie

dane wejściowe: posortowana lista elementów L, szukany element x

wynik: indeks elementu x, lub -1 jeśli elementu nie ma na liście

Wyszukiwanie binarne:

1) podziel tablicę na dwie połowy.
2) jeśli szukany element jest mniejszy niż początek drugiej połowy, szukaj w pierwszej połowie.
3) jeśli element jest większy niż początek drugiej połowy, szukaj w drugiej połowie
4) wróć do punktu 1 biorąc odpowiednią połowę listy

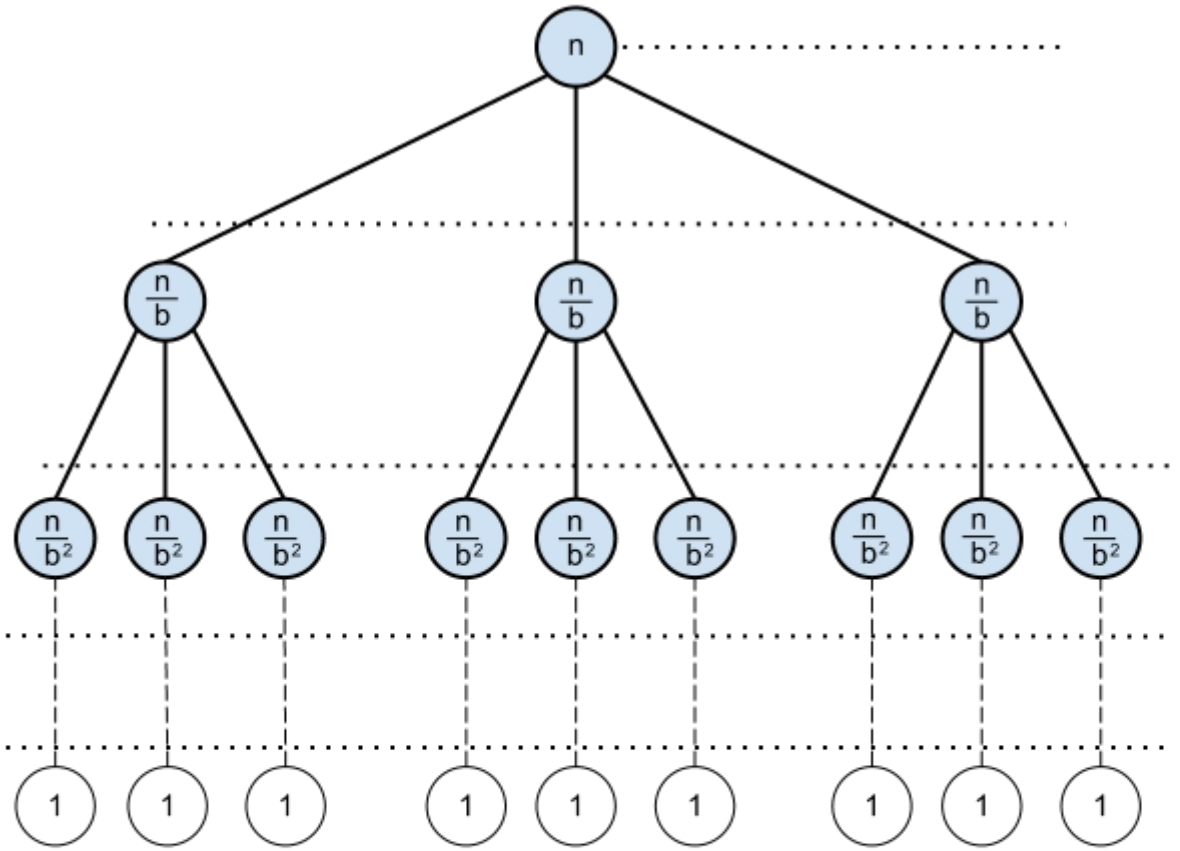


http://www.algorytm.org/dla-poczatkujacych/szukanie-polowkowe-binarne.html

Zadanie algorytmiczne: wyszukiwanie

dane wejściowe: posortowana lista elementów L, szukany element x

wynik: indeks elementu x, lub -1 jeśli elementu nie ma na liście



Złożoność obliczeniowa:
liczba podziałów "na pół":

$N/2^x = 1$

$N = 2^x$

$X = \log_2 N$

Czyli dla wyszukiwania binarnego złożoność obliczeniowa to O(logN)

Zadanie algorytmiczne: wyszukiwanie

dane wejściowe: (nie)posortowana lista elementów L, szukany element x

wynik: indeks elementu x, lub -1 jeśli elementu nie ma na liście

Złożoność obliczeniowa:

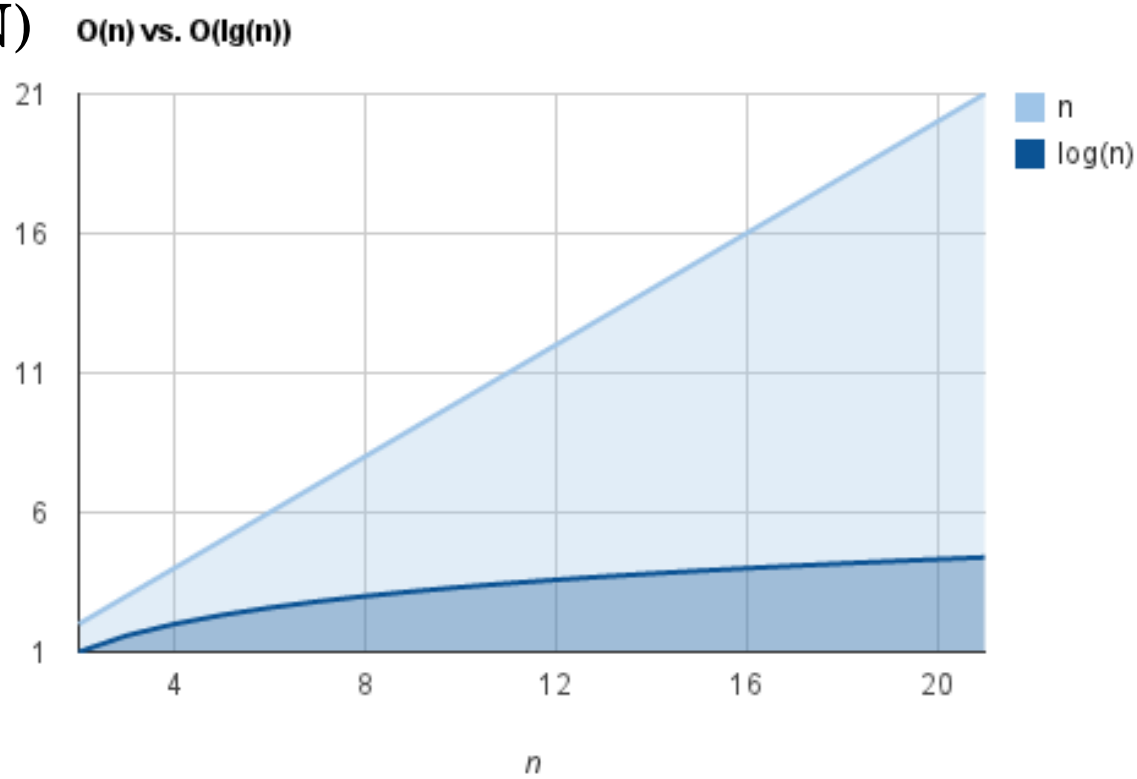
wyszukiwanie sekwencyjne – O(N)

Wyszukiwanie binarne – O(logN)

zysk dla N=1000

1000/log(1000) = 150

wartość przybliżona!



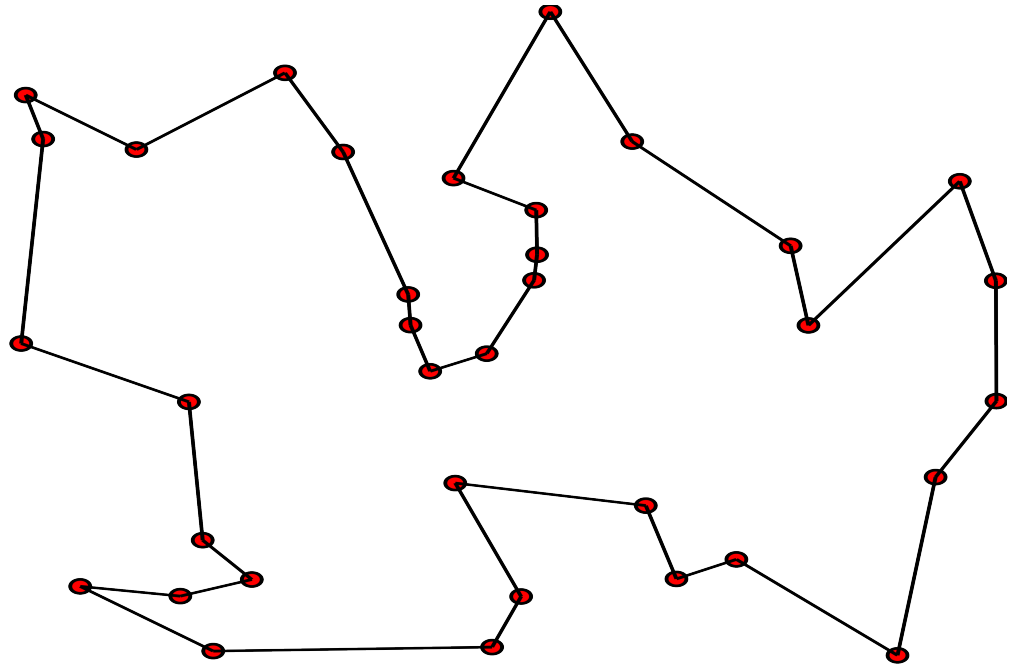
http://www.stoimen.com/blog/2012/07/03/computer-algorithms-balancing-a-binary-search-tree/

Zadanie algorytmiczne: problem komiwojażera

dane wejściowe: lista „miast”, tabela odległości między każdą parą miast.

wynik: najkrótsza ścieżka łącząca wszystkie miasta, każde z miast tylko raz

wariant decyzyjny: odpowiedź na pytanie “Czy istnieje ścieżka o długości mniejszej niż X”, gdzie X należy do danych wejściowych.



Zadanie algorytmiczne: problem komiwojażera

Algorytm siłowy (*ang. brute force*): sprawdzamy wszystkie możliwe ścieżki, dla każdej liczymy długość, wybieramy najkrótszą.

Złożoność obliczeniowa:

Możliwe ścieżki:

1 2 3 4 5 6 7 równoważna 7 6 5 4 3 2 1 (kierunek obiegu nie jest

1 3 2 4 5 6 7 równoważna 7 6 5 4 2 3 1 istotny, czyli po dwie równoważne drogi)

...

Ponieważ ścieżka jest zamknięta to droga

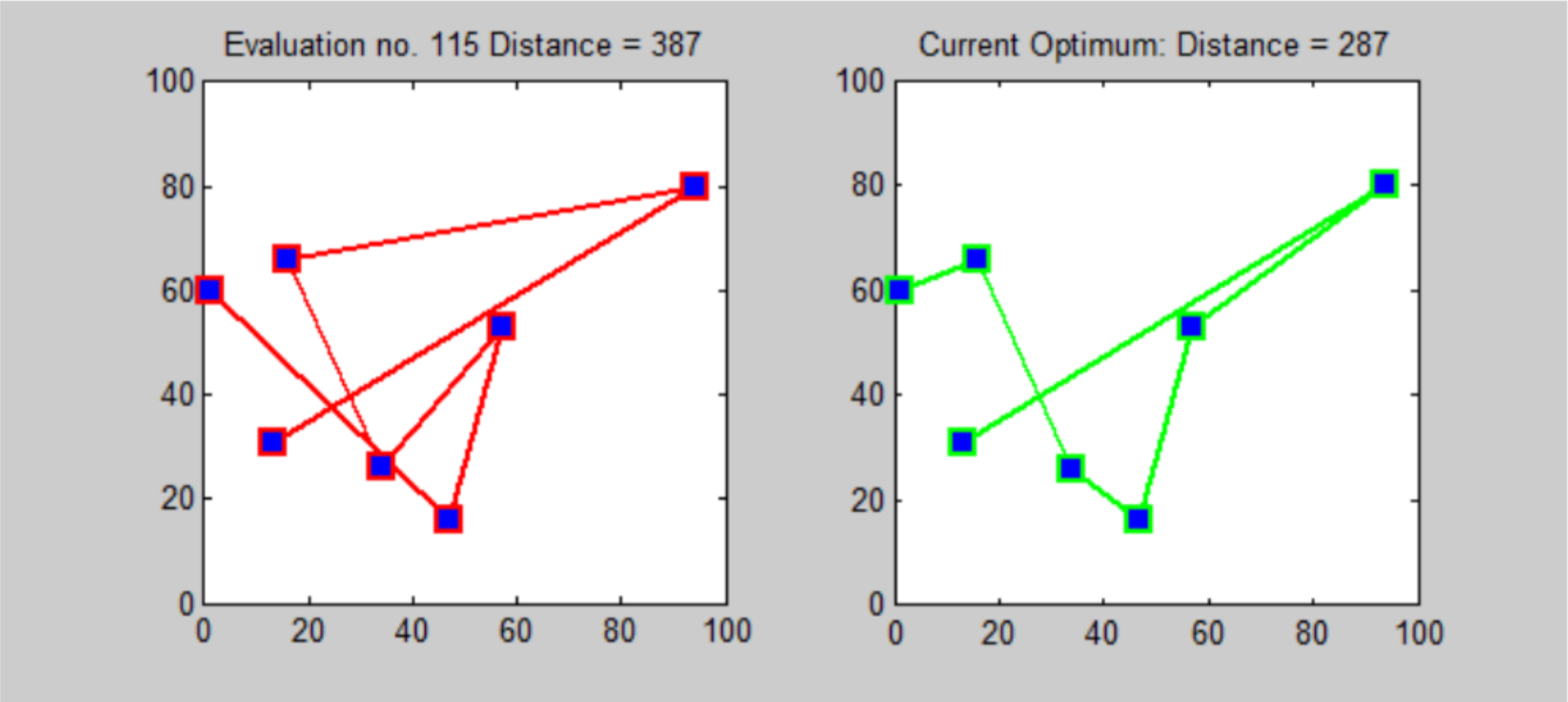
1 2 3 4 5 6 7 jest równoważna 7 1 2 3 4 5 6 itd. (N możliwych początków, czyli N dróg równoważnych)

Liczba nierównoważnych ścieżek: (N-1)!/2

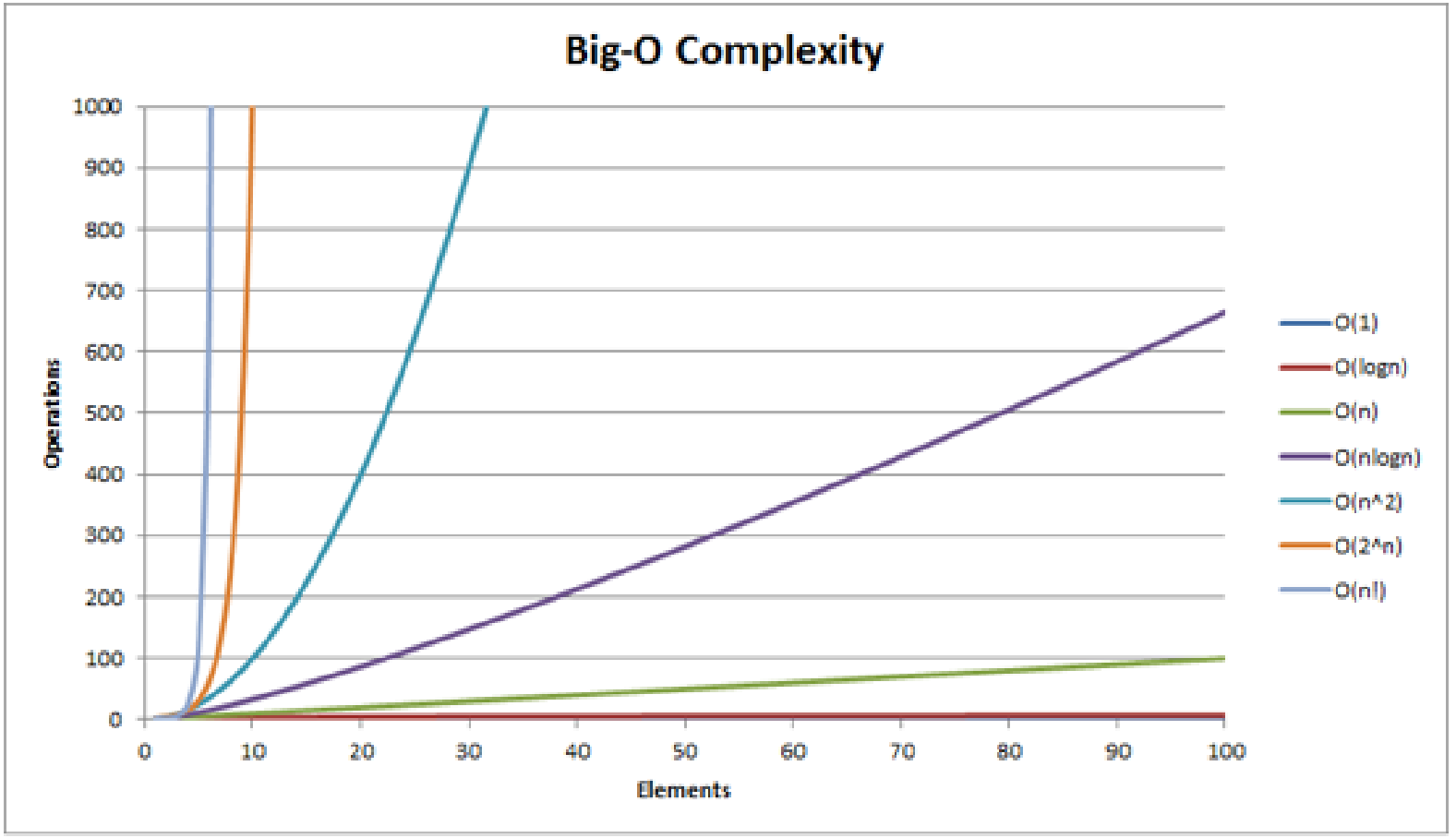
Złożoność obliczeniowa tego algorytmu to O(N!)

Zadanie algorytmiczne: problem komiwojażera

Algorytm siłowy (*ang. brute force*): sprawdzamy wszystkie możliwe ścieżki, dla każdej liczymy długość, wybieramy najkrótszą.



Zadanie algorytmiczne: problem komiwojażera



http://www.daveperrett.com/articles/2010/12/07/comp-sci-101-big-o-notation/

Złożoność obliczeniowa O(N!): już przy 20 miastach trzeba wykonać 2432902008176640000 = 2.4×10^{18} operacji.

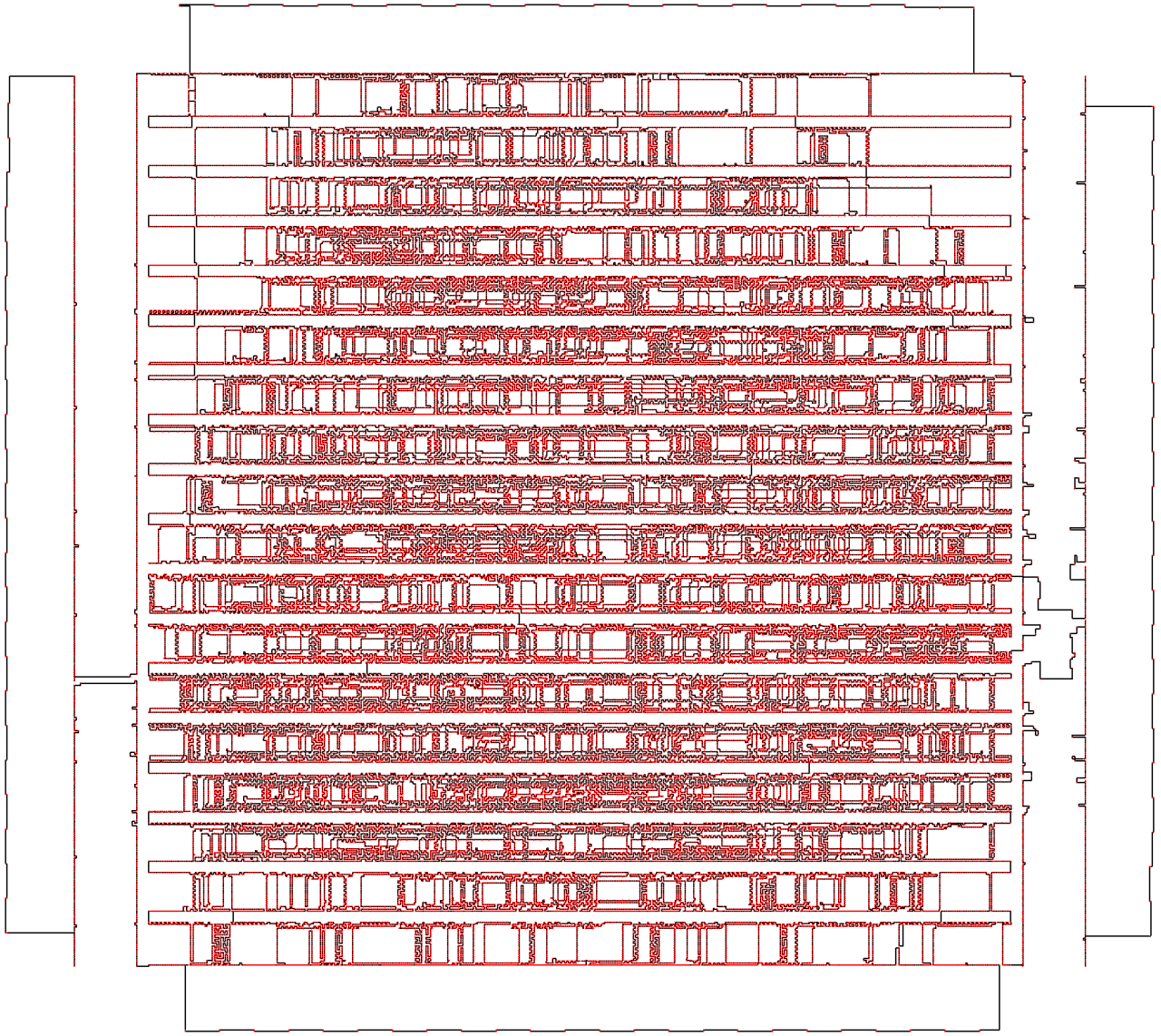
Zakładając, że komputer sprawdza 10^6 dróg na sekundę sprawdzenie 20 miast zajęło by 2.5×10^{12} sekund, czyli 76 000 lat.

Zadanie algorytmiczne: problem komiwojażera

Największe znane rozwiązanie (2006 rok):
85900 “miast”. Rozwiązanie użyte do planowania drogi lasera produkującego układy scalone.

Ścisłe algorytmy inne niż siłowy mają złożoność $O(N^2 2^N)$

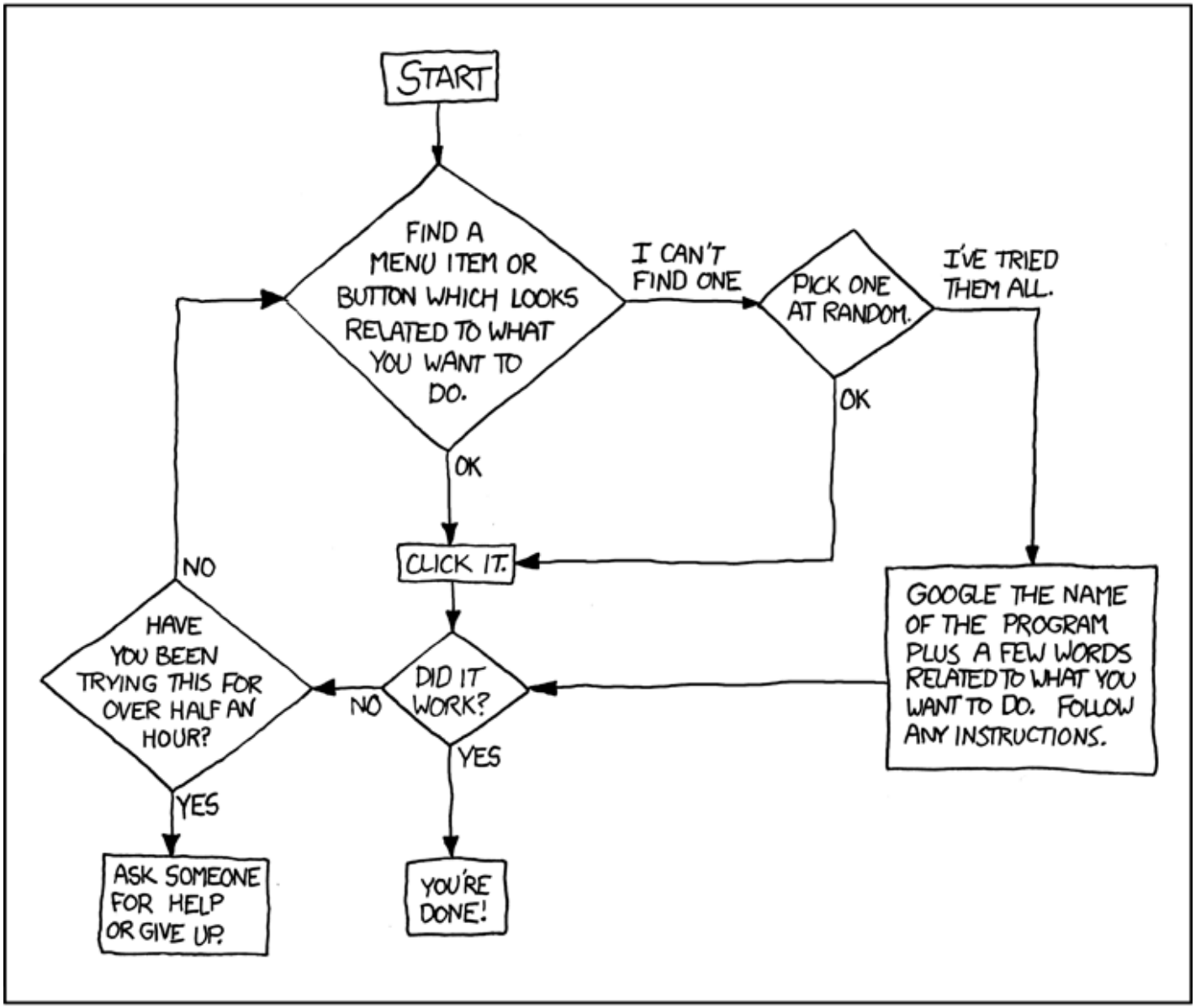
Szybsze rozwiązania można uzyskać metodami przybliżonymi, które dają rozwiązania “bliskie” optymalnemu.



http://www.math.uwaterloo.ca/tsp/pla85900/index.html

DEAR VARIOUS PARENTS, GRANDPARENTS, CO-WORKERS, AND OTHER "NOT COMPUTER PEOPLE."

WE DON'T MAGICALLY KNOW HOW TO DO EVERYTHING IN EVERY PROGRAM. WHEN WE HELP YOU, WE'RE USUALLY JUST DOING THIS:



PLEASE PRINT THIS FLOWCHART OUT AND TAPE IT NEAR YOUR SCREEN. CONGRATULATIONS; YOU'RE NOW THE LOCAL COMPUTER EXPERT!

http://xkcd.com/627/

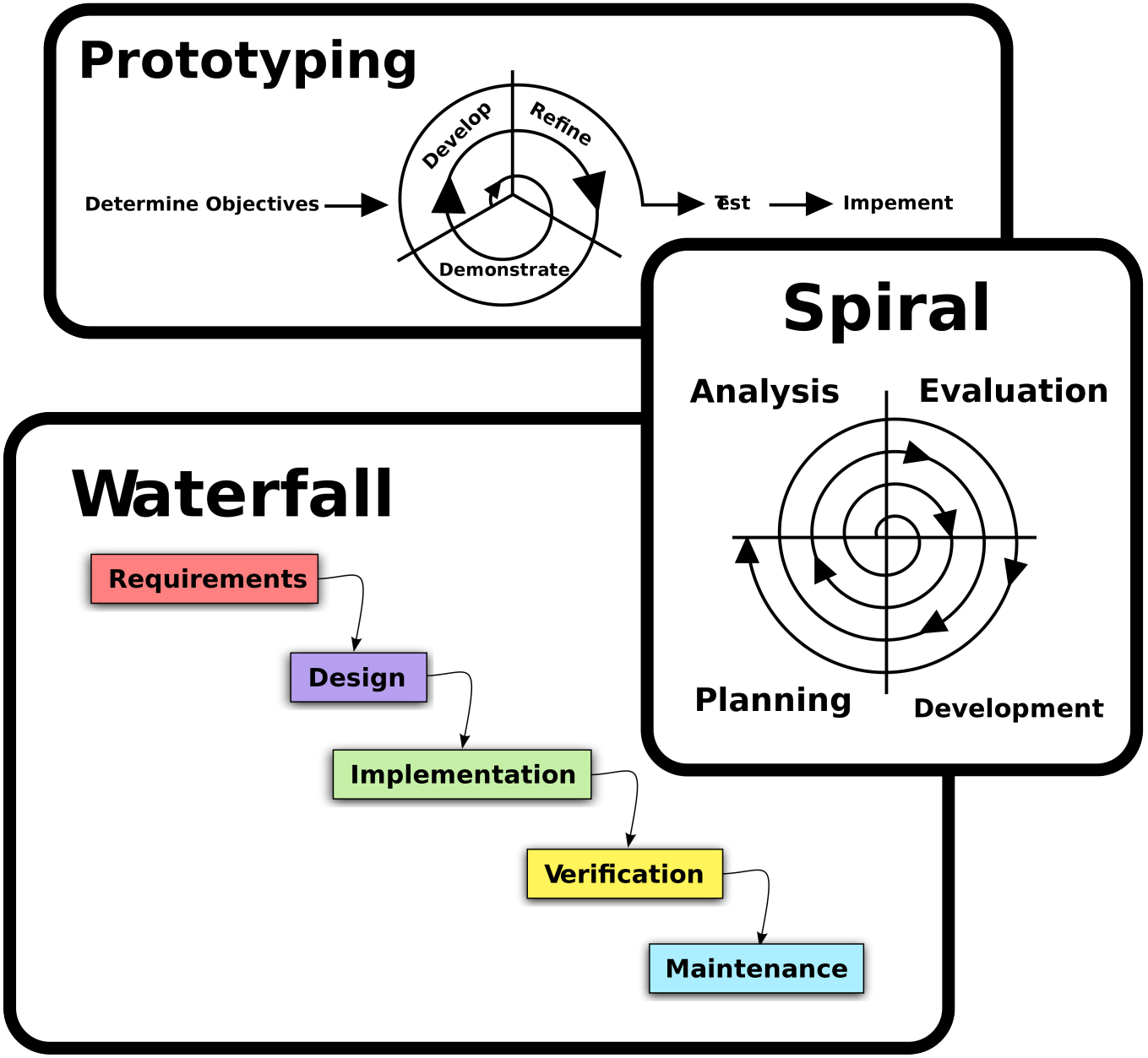
Języki programowania

Język programowania: zestaw symboli i gramatyka używane do tworzenia programów komputerowych.
Języki wysokiego poziomu (np. C++, Python, Java) są skonstruowane tak, by ułatwić zrozumienie programu (kodu).
Języki niskiego poziomu (np. assembler) używają konstrukcji możliwie najbliższych do wykonywanych rzeczywiście przez procesor.

```
fib:
    mov edx, [esp+8]
    cmp edx, 0
    ja @f
    mov eax, 0
    ret

    @@:
    cmp edx, 2
    ja @f
    mov eax, 1
    ret

    @@:
    push ebx
    mov ebx, 1
    mov ecx, 1

    @@:
```

```
unsigned int fib(unsigned int n)
{
    if (n <= 0)
        return 0;
    else if (n <= 2)
        return 1;
    else {
        int a,b,c;
        a = 1;
        b = 1;
        while (1) {
            c = a + b;
            if (n <= 3) return c;
            a = b;
            b = c;
            n--;
        }
    }
}
```

Projektowanie programów



Kompilacja

Kompilacja programu (kodu): proces tłumaczenia kodu napisanego w wybranym języku na kod maszynowy, zrozumiały dla procesora. Wynikiem kompilacji jest program, który można uruchomić. W czasie kompilacji jest analizowany cały kod, co oznacza, że nie może w nim być nigdzie błędów językowych.

Interpretacja: wykonywanie kodu programu krok po kroku, bez analizy poprawności językowej całości. Interpretacja jest prowadzona do napotkania błędu w kodzie, lub poprawnego zakończenia działania programu.

Języki kompilowane: C++

Języki interpretowane: PHP, Java, Python

Wszystkie przykłady na następnych slajdach pochodzą ze strony http://brain.fuw.edu.pl/edu/TI:Programowanie_z_Pythonem

Język Python

Struktury danych: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

zmienna: ”miejsce” na konkretne dane. Zmienna ma swoją nazwę, typ i wartość.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Nazwa pliku: wciecia.py

i = 5
 print 'Wartość zmiennej to ', i # Błąd! Zauważ spację na początku linii
print 'Powtarzam, wartość zmiennej to ', i
```

```
 File "wciecia.py", line 4
   print 'Wartość zmiennej to ', i # Błąd! Zauważ spację na początku linii
   ^
IndentationError: unexpected indent
```

Język Python

Struktury danych: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

zmienna: ”miejsce” na konkretne dane. Zmienna ma swoją nazwę, typ i wartość.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Nazwa pliku: zmienne.py
i = 5
print i
i = i + 1
print i
s = '''To jest napis wielolinijkowy.
To jest drugi wiersz.'''
print s
```

```
$ python zmienne.py
5
6
To jest napis wielolinijkowy.
To jest drugi wiersz.
```

Język Python

Wyrażenia: operacje na zmiennych, „wzory”.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Nazwa pliku: wyrazenia.py

dlugosc = 5
szerokosc = 2
pole = dlugosc * szerokosc
print 'Pole wynosi', pole
print 'Obwód wynosi', 2 * (dlugosc + szerokosc)
```

```
$ python wyrazenia.py
Pole wynosi 10
Obwód wynosi 14
```

Język Python

Struktury danych: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

lista (macierz jednowymiarowa): zestaw podstawowych elementów, np. liczb.

```
a = [2, 3, 4, 5]
print 'sekwencja a:', a
```

```
a = range(2,6)
print 'sekwencja a:', a
```

```
$ python for.py
sekwencja: [2, 3, 4, 5]
```

Język Python

Struktury sterujące: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

”jeśli Q”: jeżeli zadanie logiczne, oznaczone symbolem Q jest prawdziwe, np: Q – „x>0”

```
if warunek:
    blok kodu,
    który ma być wykonany
    jeśli warunek jest prawdziwy
```

Język Python

Struktury sterujące: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

”jeśli Q”: jeżeli zadanie logiczne, oznaczone symbolem Q jest prawdziwe, np: Q – „x>0”

```
if warunek:
    blok kodu,
    który ma być wykonany
    jeśli warunek jest prawdziwy
else:
    blok kodu,
    który ma być wykonany
    jeśli warunek jest fałszywy
```

Język Python

Struktury sterujące: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

”jeśli Q”: jeżeli zadanie logiczne, oznaczone symbolem Q jest prawdziwe, np: Q – „x>0”

```
if warunek1:
    blok kodu,
    który ma być wykonany
    jeśli warunek1 jest prawdziwy
elif warunek2:
    blok kodu,
    który ma być wykonany
    jeśli warunek2 jest prawdziwy
else:
    blok kodu,
    który ma być wykonany
    jeśli każdy z powyższych warunków jest fałszywy
```

Lista obecności

Język Python

Struktury sterujące: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

”jeśli Q”: jeżeli zadanie logiczne, oznaczone symbolem Q jest prawdziwe, np: Q – „x>0”

bezpośrednie następstwo: “wykonaj A potem B”

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
temperatura = 11

if temperatura > 0:
   print "jest ciepło!"
else:
   print "uwaga, przymrozek!"
```

Język Python

Struktury sterujące: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

iteracja ograniczona: "wykonaj A dokładnie N razy"

```
for zmienna in sekwencja:
    blok
```

```
a = [2, 3, 4, 5]
print 'sekwencja a:', a
```

```
a = range(2,6)
print 'sekwencja a:', a
```

Język Python

Struktury sterujące: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

iteracja ograniczona: "wykonaj A dokładnie N razy"

```
# cała sekwencja na raz
print 'sekwencja:', [2, 3, 4, 5]

# sekwencja element po elemencie
for i in [2, 3, 4, 5]:
    print 'element sekwencji:', i
```

```
$ python for.py
sekwencja: [2, 3, 4, 5]
element sekwencji: 2
element sekwencji: 3
element sekwencji: 4
element sekwencji: 5
```

Język Python

Struktury sterujące: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

iteracja warunkowa: “dopóki Q, wykonuj A”

```
while warunek:
    blok
```

Język Python

Struktury sterujące: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

iteracja warunkowa: "dopóki Q, wykonuj A"

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

liczba = 23
dzialaj = True
while dzialaj:
    strzal = int(raw_input('Wpisz liczbę całkowitą: '))
    if strzal == liczba:
        print 'Gratulacje, zgadłeś ją!'
        print '(Ale nic nie wygrałeś.)'
        dzialaj = False  # To sprawia, że warunek przestaje być
                         # spełniony
    elif strzal < liczba:
        print 'Nie, szukana liczba jest większa od podanej.'
    else:
        print 'Nie, szukana liczba jest mniejsza od podanej.'
print 'Koniec programu.'
```

Język Python

Funkcje: elementy algorytmu wykonujące określone obliczenia dla zadanych danych wejściowych (argumentów).
Często bloki programu wykonujące jakąś czynność są zamykane w funkcje, które nie wymagają żadnych argumentów

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Nazwa pliku: function1.py

# początek definicji funkcji
def przywitajSie():
    print 'Sie ma!' # instrukcje tworzące funkcję
# koniec definicji funkcji

przywitajSie() # wywołanie (użycie) funkcji
przywitajSie() # ponowne wywołanie funkcji
```

Język Python

Funkcje: elementy algorytmu wykonujące określone obliczenia dla zadanych danych wejściowych (argumentów).

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Nazwa pliku: func_param.py

def printMax(a, b):
    if a > b:
        print a, 'jest większe'
    elif a == b:
        print a, 'jest równe', b
    else:
        print b, 'jest większe'

printMax(3, 4) # bezpośrednio podane wartości

x = 5
y = 7

printMax(x, y) # zmienne podane jako argumenty
```

Język Python

Funkcje: elementy algorytmu wykonujące określone obliczenia dla zadanych danych wejściowych (argumentów).

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Nazwa pliku: continue.py

while True:
    s = raw_input('Wpisz coś: ')
    if s == 'quit':
        break
    if len(s) < 3:
        print 'Za krótkie.'
        continue
    print 'Wpis jest wystarczającej długości.'
    print 'Liczba słów wpisu:', len(s.split()), '.'
```

Język Python

Funkcje: elementy algorytmu wykonujące określone obliczenia dla zadanych danych wejściowych (argumentów).

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Nazwa pliku: func_return.py

def maksimum(x, y):
    if x > y:
        return x
    else:
        return y

print maksimum(2, 3)
```

Język Python

Wejście/wyjście: ”interface” dostarczania danych do programu i przekazywania wyników na zewnątrz programu.

```
licba_wprowadzona =  int(raw_input('Podaj liczbę: ' ) )
```

```
    s = raw_input('Wpisz coś: ')
```

Materiały

- Jerzy Mieścicki, Wstęp do informatyki nie tylko dla informatyków
- Dawid Harel, Rzecz o istocie informatyki. Algorytmika
- http://brain.fuw.edu.pl/edu/TI:Programowanie_z_Pythonem