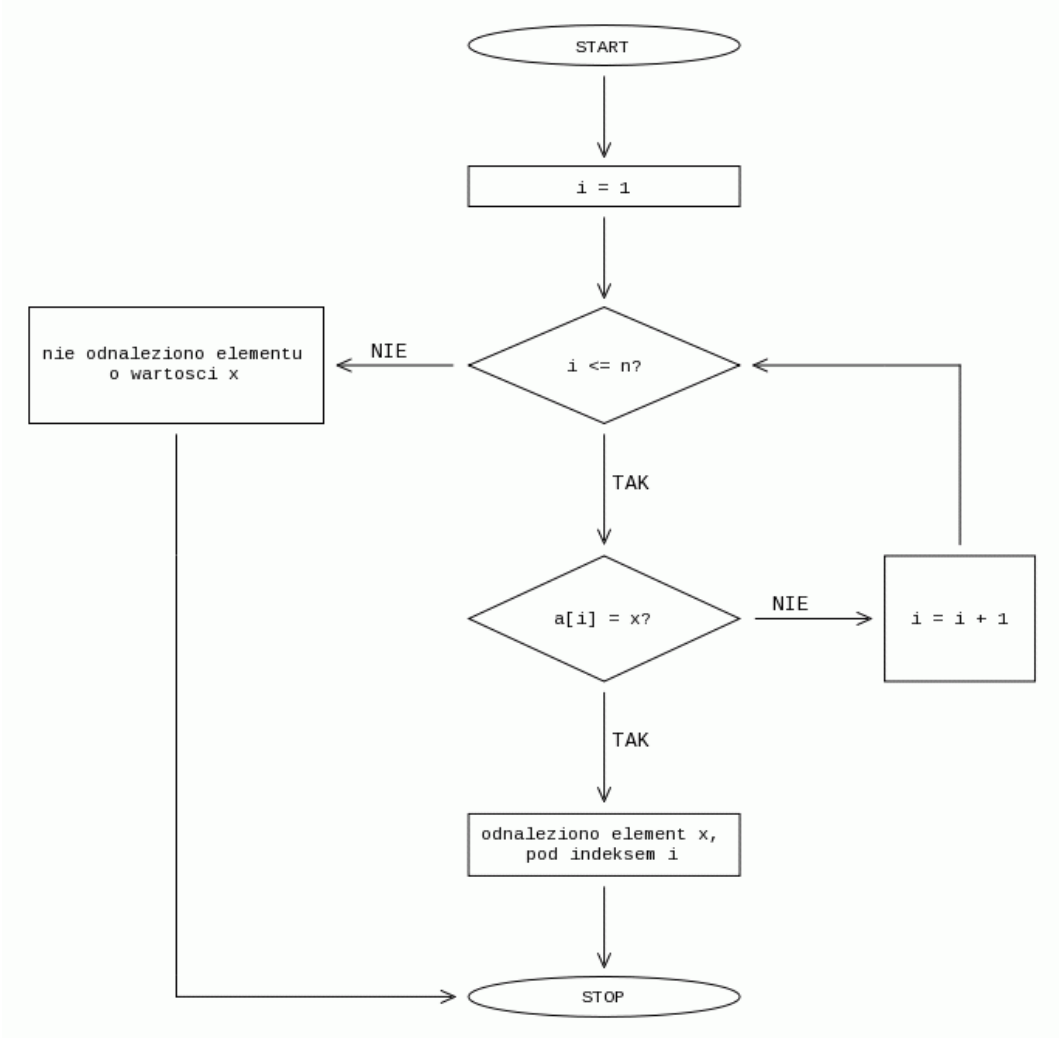
Szukanie elementów z wartownikiem

Materiały ze strony: algorytm.org

Załóżmy że mamy daną tablicę *n*-elementów i chcemy odnaleźć w niej zadany element *x*. Niech będzie to tablica *a* o indeksach od 1 do *n*. Czyli kolejne jej elementy oznaczymy: *a[1], a[2], a[3], ..., a[n-1], a[n]*.
Jeżeli przyjrzymy się dokładnie klasycznemu algorytmowi przeglądania tablicy w poszukiwaniu elementu:



zauważymy, że dla każdego elementu wykonywane są dwa porównania:

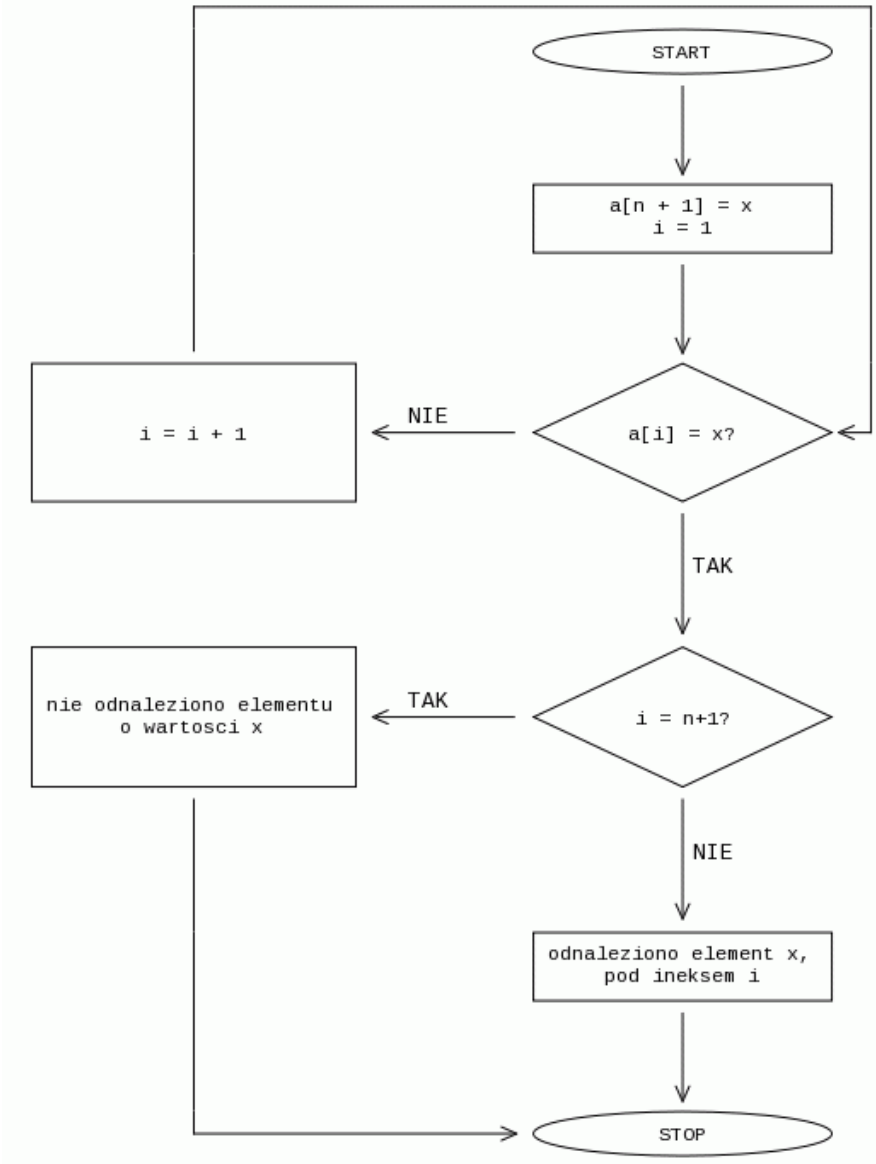
- pierwsze: czy znaleźliśmy się już na końcu tablicy,
- drugie: czy aktualnie przeglądany element jest równy poszukiwanemu.

Liczbę porównań można zredukować wykorzystując algorytm wyszukiwania z wartownikiem. Nazwa tego algorytmu bierze się ze sposobu, w jaki wykorzystywany jest element szukany *x*.

By odnaleźć element *x* podejmiemy następujące kroki:

- na końcu tablicy (czyli pod indeksem *n+1*) wstawimy szukany element *x* - będzie to nasz wartownik, w przypadku gdy nie znajdziemy go nigdzie indziej w tablicy, zabezpieczy nas on przed wyjściem poza tablicę,
- przejdziemy po kolejnych elementach tablicy, tak długo aż nie znajdziemy szukanego elementu,
- w momencie znalezienia szukanego elementu *x* sprawdzamy, który jest to element tablicy? Jeżeli jest to ostatni element tablicy (*n+1*) to trafiliśmy na naszego wartownika i oznacza to, że w tablicy nie było szukanego elementu *x*, w przeciwnym razie element *x* został odnaleziony.

W stosunku do klasycznego algorytmu wyszukiwania oszczędzamy czas na sprawdzaniu czy osiągnęliśmy koniec tablicy. Sprawdzenie to występuje raz - po odnalezieniu elementu szukanego, a nie tak jak poprzednio przed każdym sprawdzeniem kolejnego elementu. Operację odnajdowania elementu w tablicy z wartownikiem możemy zapisać następującym schematem blokowym:



Algorytm ten mimo, że jest szybszy od **zwykłego wyszukiwania** to ma taką samą jak on złożoność obliczeniową wynoszącą *O(n)*, co oznacza, że czas potrzebny na wyszukanie

elementu tą metodą rośnie w sposób liniowy wraz z liniowym wzrostem liczby elementów w przeszukiwanej tablicy. To znaczy, że czas potrzebny na wyszukanie dwa razy większej liczby elementów wzrośnie dwukrotnie.
Podczas implementowania tego rozwiązania należy zapewnić miejsce w tablicy potrzebne do dodania wartownika. Należy więc deklarować tablicę o rozmiarze o jeden większym jak liczba przechowywanych elementów.

Przykład implementacji w języku Python

```
#szukanie z wartownikiem
#www.algorytm.org

from random import randint

ileliczb = 20           #rozmiar tablicy
mina = 0                    #minimalna wartosc elementu
maxa = 100                  #maksymalna wartosc elementu
straz = ""
tab = []

#wypelnij tablice przypadkowymi wartosciami

for i in range(ileliczb):
  tab.append(randint(mina, maxa))

while True:
  try:
    szukana = int(input("Podaj liczbę: "))              #odgadnij liczbe
  except ValueError:
    print("To nie jest liczba. Spróbuj jeszcze raz.")           #nie podano liczby
  else:
    tab.append(szukana)         #dodaj wartownika do tablicy
    i = 0
    while tab[i] != szukana:        #dopoki nie znaleziono elementu
      i += 1                            #przesuwaj sie dalej
    break

if i == ileliczb:
  print("Strażnik. Liczby nie ma w tablicy")    #znaleziono wartownika, szukana liczba nie
                                                #znajduje sie w tablicy
else:
  print("Znaleziono: ", str(szukana), "na miejscu: ", i)  #znaleziono podana liczbe na pozycji i
```