1. Wprowadzenie

Algorytm wyszukiwania binarnego szuka danego elementu w posortowanej tablicy. W wersji zaprezentowanej w tej pracy zwraca on index liczby w tablicy bądź -1, gdy dana liczba nie znajduje się w tablicy. Algorytm realizowany jest metodą "dziel I zwyciężaj". W praktyce dzieli on tablice na mniejsze części do chwili, gdy znajdzie on szukany element i zwróci jego index.

2. Opis działania

Działanie algorytmu opiera się na bisekcji. Poprzez podział na pomniejsze podtablice dochodzimy do momentu, w którym szukana liczba jest na środku podtablicy, bądź gdy podtablica ma jeden element i jest to szukana liczba. Szukanie liczby w tablicy polega na wyznaczeniu środka tablicy i porównania go z szukaną liczbą. Jeśli liczba w środku tablicy jest równa szukanej liczbie, wtedy zwracamy index środka. W razie, gdy liczba znajdująca się na środku tablicy jest większa od szukanej liczby, przechodzimy do analogicznego przeszukiwania lewej strony tablicy, jeżeli liczba szukana jest większa, przechodzimy do analogicznego przeszukiwania prawej strony tablicy. Algorytm charakteryzuje się logarytmiczną złożonością obliczeniową.

3. Wizualizacja

https://www.youtube.com/watch?v=ktRgmxvcR3I

Film przedstawiający wizualizację wyszukiwania binarnego

4. Przykładowy kod w języku Python3 (wersja iteracyjna)

```
# Wersja iteracyjna
def wyszukiwanie_binarne(tab, lewy, prawy, szukany):

    while lewy < prawy: # Dopóki podtablica ma prawo istnieć

        # Wyznaczamy środek szukanego obszaru
        srodek = (lewy+prawy) // 2

        # Jeżeli znaleźliśmy szukaną liczbę
        if tab[srodek] == szukany:
            return srodek

        # Jeżeli szukana liczba znajduje się w prawej części
        elif szukany > tab[srodek]:
            lewy = srodek + 1

        # Jeżeli szukana liczba znajduje się w lewej części
        elif szukany < tab[srodek]:
            prawy = srodek

    return –1
```

5. Przykładowy kod w języku Python3 (wersja rekurencyjna)

```
# Wersja rekurencyjna 1
def wyszukiwanie_binarne_1(tab, lewy, prawy, szukany):

    if prawy > lewy:

        srodek = lewy + ((prawy-lewy) // 2) # Wyznaczanie środka
```

```
    if srodek > len(tab) - 1:
        return -1

    # Jeśli szukany jest na środku danego przedziału
    if tab[srodek] == szukany:
        return srodek

    # Jeśli element szukany jest mniejszy od środka (znajduje się po lewej stronie)
    elif tab[srodek] > szukany:
        return wyszukiwanie_binarne_1(tab, lewy, srodek, szukany)

    # Jeśli element szukany jest większy od środka (znajduje się po prawej stronie)
    else:
        return wyszukiwanie_binarne_1(tab, srodek+1, prawy, szukany)

else:
    return –1
```

6. Przykładowy kod w języku Python3 (rekurencyjna kod specyficzny dla pythona, oparty na wielu tablicach)

```
# Wersja rekurencyjna 2
def wyszukiwanie_binarne_2(tab, szukany, ind=0):
    srodek = (len(tab)-1) // 2 # Wyznaczanie środka

    if len(tab) == 1 and tab[0] != szukany: # Gdy podtablica ma jeden element różny od szukanej liczby, to znaczy, że dan
a liczba nie znajduje się w tablicy
        return -1

    elif tab[srodek] == szukany: # Jeśli liczba równa jest szukanej liczbie
        return  ind + srodek
```

```
elif tab[srodek] < szukany: # Jeśli szukana jest na prawo od środka
    return wyszukiwanie_binarne_2(tab[srodek+1:], szukany, ind+srodek+1)

else: # Jeśli szukana jest na lewo od środka
    return wyszukiwanie_binarne_2(tab[:srodek], szukany, ind)
```