**1.Лінійний пошук**

**Ідея**

Проглядати почергово елементи масиву, доки не буде знайдено шуканий елемент або не буде досягнуто кінець масиву.

**Псевдокод**

*i*=0   
**поки** (*i* < *N*  **і** *a*[*i*] != *Key*)    
   { *i*=*i*+1}   
**якщо** *i*=*N*  
   **то** елемент відсутній  
**інакше** елемент має номер*і*

Якщо відрізок має довжину N, то знайти рішення з точністю до {\displaystyle \epsilon } можна за час {\displaystyle N \over \epsilon }. Таким чином асимптоматична складність алгоритму - {\displaystyle O(n)}. У зв'язку з малою ефективністю в порівнянні з іншими алгоритмами лінійний пошук зазвичай використовують лише тоді, коли відрізок пошукової системи містить дуже мало елементів, однак лінійний пошук не вимагає додаткової пам'яті або обробки/аналізу функції, так що може працювати в потоковому режимі при безпосередньому отриманні даних з будь-якого джерела. Так само, лінійний пошук часто використовується у вигляді лінійних алгоритмів пошуку максимуму/мінімуму.

Це самий простий вид пошуку деякого елемента серед інших, що здійснюється за допомогою перевірки кожного елемента до тих пір, поки вони не будуть збігатися. Загальна ідея цього виду пошуку така: усі елементи розглядаються послідовно, один за одним. Це дає змогу не пропустити жодного елемента. Якщо збіг буде знайдено, то пошук припиняється і його результат є позитивним. Якщо не знайдено, то результат буде негативним.

2.

Перевагами такого пошуку є простота його реалізації, він не потребує додаткового об’єму пам’яті або додаткової роботи з функціями. Це дозволяє працювати вже під час отримання даних.

Також існує певний покращений послідовний алгоритм, який пришвидшує пошук. У множині встановлюється бар’єр, тобто елемент, який задовольняє пошуку. У циклі відпадає необхідність перевірки умови, зв’язаної з границями множини. Таким чином буде обмежена зміна індексу.

3. Ідея алгоритму бінарного пошуку така:

• порівняти аргумент пошуку ключ зі значенням середнього елемента *х [середовищ]*масиву *х,*

де середовищ = *[П / 2],*а [з] ціла частина числа с

• якщо вони рівні, то пошук завершено, інакше, якщо ключ < *х [середовищ],*виконати аналогічним чином пошук в позиціях масиву*,* попередніх позиції середовищ, в іншому випадку, якщо ключ> *х [середовищ],*виконати аналогічним чином пошук в позиціях масиву *х*, Наступних за позицією середовищ

Виключити з подальшого розгляду частина масиву дозволяє той факт, що масив впорядкований

Кожне порівняння зменшує число можливих кандидатів у 2 рази Максимальне число кроків пошуку буде в тому випадку, коли аргумент пошуку знаходиться на початку або в кінці масиву У цьому випадку буде потрібно log2n + 1 ітерацій Дійсно, якщо число елементів у масиві дорівнює *п = 2т, ключ*буде знайдений, коли нерозглянутим залишиться тільки один елемент, тобто через *т*кроків У свою чергу, при заданому *п*маємо *т*= log2*n* Після аналізу останнього елемента отримуємо загальне число ітерацій log2*n*+ 1 Тому обчислювальна складність бінарного пошуку складає O (log2*n*)

Однак наведений алгоритм не дозволяє в загальному випадку точно вирішити завдання пошуку, коли файл або масив містять повторювані значення ключів

4. Послідовний пошук не гарантує знаходження елементу. Також і бінарний. Тільки лінійний пошук з барєром гарантує знаходження елементу,.

5. Лінійний пошук:

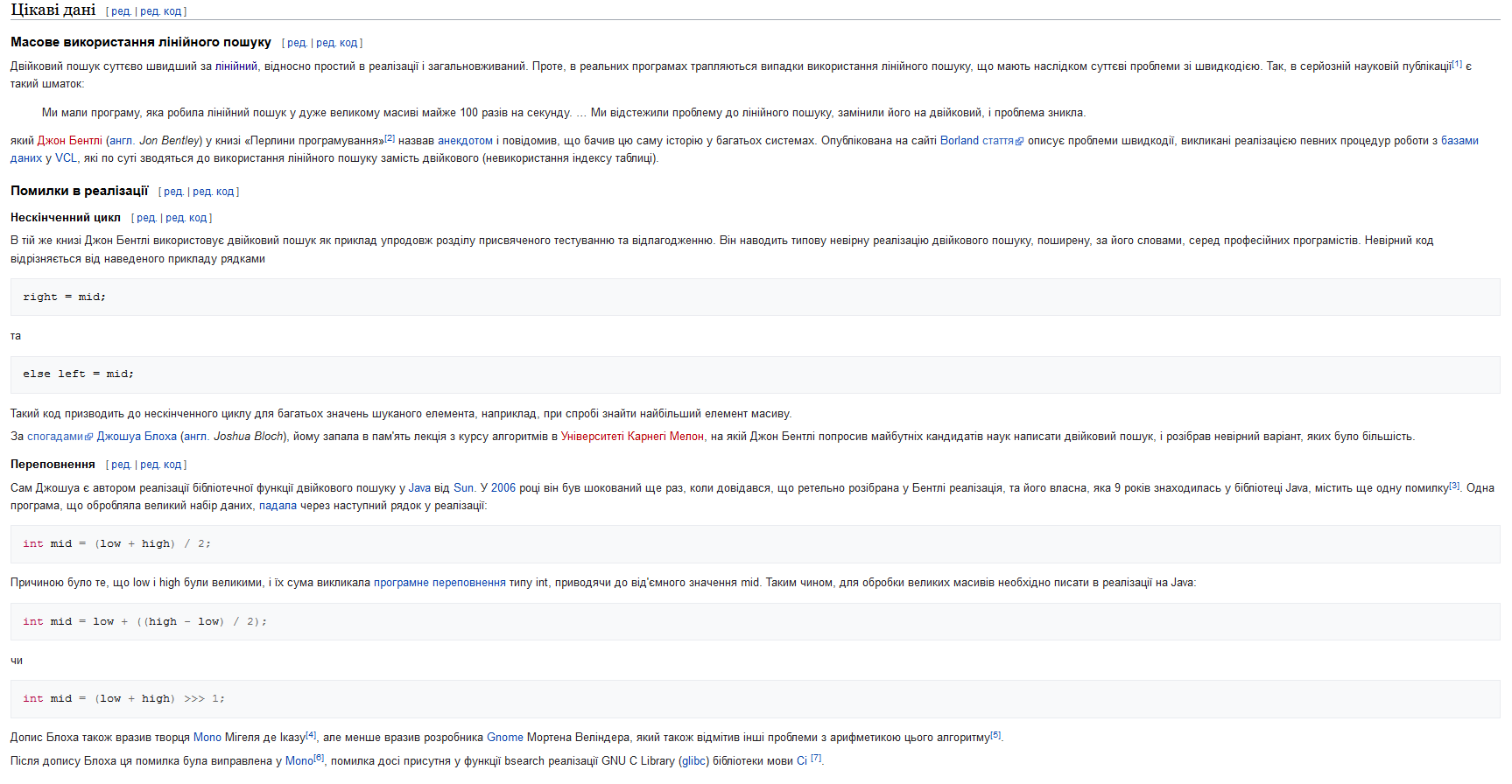
**Первага:** лінійний пошук не вимагає додаткової пам'яті або обробки/аналізу функції, так що може працювати в потоковому режимі при безпосередньому отриманні даних з будь-якого джерела. Так само, лінійний пошук часто використовується у вигляді лінійних алгоритмів пошуку максимуму/мінімуму.

**Недолік:** У зв'язку з малою ефективністю в порівнянні з іншими алгоритмами лінійний пошук зазвичай використовують лише тоді, коли відрізок пошукової системи містить дуже мало елементів

**Бінарний пошук:**

**Переваги:** Суттєво швидший за лінійний пошук

**Недоліки:** Працює тільки на впорядкованих множингах.



**5**