KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Algoritmų sudarymas ir analizė

1 Laboratorinis darbas

Atliko:

IF 8/1 grupės stud.
Martynas Kemežys

Priėmė
doc. Dalius
Makackas

KAUNAS, 2020

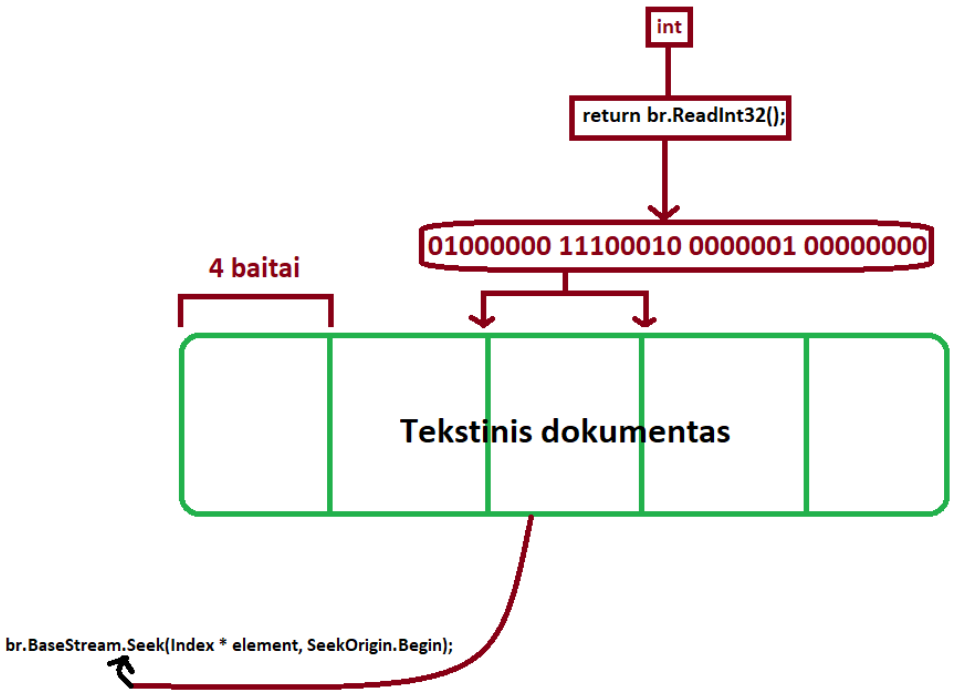
TURINYS

1. Užduotis

Palyginkite rikiavimo algoritmus, kai rikiavimas atliekamas masyve ir sąraše („Linked List“), t. y. galimos tik tai struktūrai būdingos operacijos. Abu šiuos algoritmus realizuokite ir palyginkite dviem atvejais: 1) rikiuojami duomenų elementai saugomi operatyviojoje atmintyje; 2) rikiuojami duomenų elementai visą laiką saugomi išorinėje (diskinėje) atmintyje, o operatyviojoje atmintyje gali būti saugojami tik neindeksuotuose kintamuosiuose (neturi būti masyvo, sąrašo ar jų analogų operatyvinėje atmintyje).Programose turi būti numatyta galimybė išvesti į ekraną nesurikiuoto ar surikiuoto masyvo ar sąrašo fragmentą nuo dėstytoju nurodyto elemento.

Variantas: 9. Rikiavimas „Insertion Sort“

2. Pasirinktos duomenų struktūros realizuotos išorinėje atmintyje struktūrinė diagrama



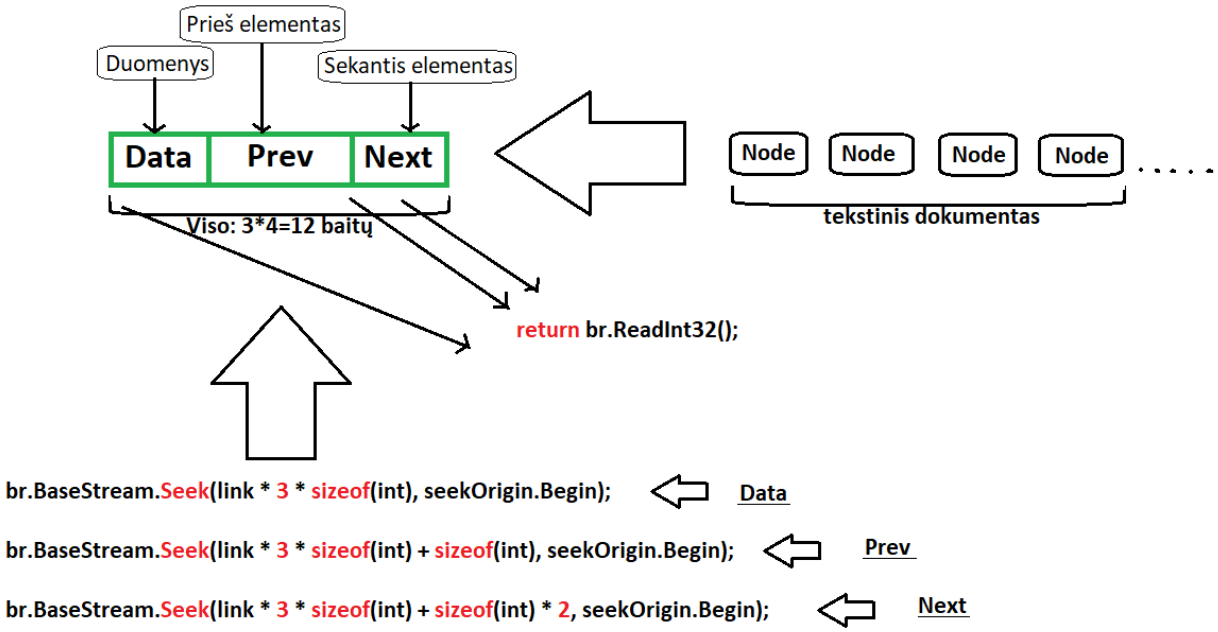
Pav. 1

Programos kodo fragmentas realizuojantis elemento nuskaitymą iš išorinės atminties:

```
public int this[int index]{
    get{
        br.BaseStream.Seek(index * element, SeekOrigin.Begin);
        return br.ReadInt32();
    }
    set{
        bw.BaseStream.Seek(index * element, SeekOrigin.Begin);
        bw.Write(value);
    }
}
```

Pav. 2

Sąrašo duomenų struktūros realizavimo išorinėje atmintyje diagrama:



Pav. 3

Programos kodo fragmentas realizuojantis elemento nuskaitymą iš išorinės atminties:

```
public int GetData(int link)
{
    br.BaseStream.Seek(link * 3 * sizeof(int), SeekOrigin.Begin);
    return br.ReadInt32();
}
```

Pav. 4

3. Algoritmo analizė

Algoritmo įvertinimai literatūroje:
Blogiausiu atveju $T(n) = O(n^2)$. Geriausiu atveju $T(n) = \Theta(n)$

https://www.bigocheatsheet.com/

InsertionSort metodo analizė:

		Kaina	Kiekis
1	`public static void InsertionSort(int[] data)`		
2	`{`		
3	`int i;`	C1	1
4	`int j;`	C2	1
5	`int key;`	C3	1
6	`for (i = 1; i < data.Length; i++)`	C4	n
7	`{`		
8	`key = data[i];`	C3	n-1
9	`j = i - 1;`	C2	n-1
10	`while (j >= 0 && data[j] > key)`	C5	$\sum_{j=2}^{n}(k)$.
11	`{`		
12	`data[j + 1] = data[j];`	C6	$\sum_{j=2}^{n}(k-1)$.
13	`j--`	C2	$\sum_{j=2}^{n}(k-1)$.
14	`}`		
15	`data[j + 1] = key;`	C6	n-1
16	`}`		
17	`}`		

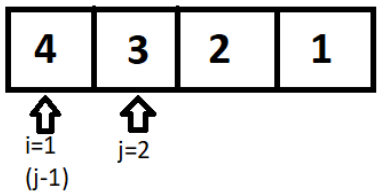
k –surasta tinkama vieta.

InsertionSort metodo sudėtingumas: $c1 * 1 + c2 * 1 + c3 * 1 + c4 * n + c3 * (n-1) + c2 * (n-1) + c5 \sum_{j=2}^{n}(k) + c6 \sum_{j=2}^{n}(k-1) + c2 \sum_{j=2}^{n}(k-1) + c6 * (n-1) = (c4 + c2 + c6 + c5)n - (c3 + c2 + c5 + c6)$

Atmetus konstantas InsertionSort metodo sudėtingumą galima laikyti **Θ(n)**

Blogiausias atvejis:
Blogiausia atvejį galime gauti tada kaip masyvas yra „surikiuotas“, bet visiškai priešingai nuo kito galo.(k=j)

$\boldsymbol{O(n^2)}$ – mes turime palyginti kiekvieną elementą. A[j] su visu rikiuotu masyvu A[1...j-1]



$$1)\ T(n) = c1 + c2 + c3 + c4n + c3(n-1) + c2(n-1) + c5\sum_{j=2}^{n}(k) + c6\sum_{j=2}^{n}(k-1) + c2\sum_{j=2}^{n}(k-1) + c6(n-1)$$

$c5\sum_{j=2}^{n}(k) = c5(\frac{n(n+1)}{2} - 1)$

$c6\sum_{j=2}^{n}(k-1) = c6(\frac{n(n-1)}{2})$

$c2\sum_{j=2}^{n}(k-1) = c2(\frac{n(n-1)}{2})$

$$2)\ T(n) = c1 + c2 + c3 + c4n + c3(n-1) + c2(n-1) + c5(\frac{n(n+1)}{2} - 1) + c6(\frac{n(n-1)}{2}) + c2(\frac{n(n-1)}{2}) + c6(n-1)$$

$$3)\ T(n) = \left(\frac{c5}{2} + \frac{c6}{2} + \frac{c2}{2}\right)n^2 + (c4 + c3 + c2 + c6)n - (c1 + c2 + c3)$$

$T(n) = an^2 + bn + c$

Vertinant asimptotiškai gauname, kad InsertionSort metodo sudėtingumas blogiausiu atveju yra $\boldsymbol{O(n^2)}$

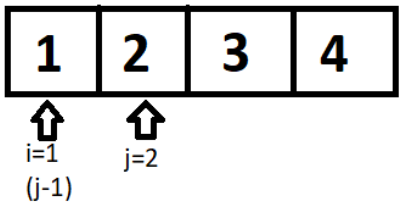
Geriausias atvejis:
Geriausią atvejį galime gauti tada kaip masyvas jau yra surikiuotas tinkamai iškarto.(k=1)

$$1)\ T(n) = c1 + c2 + c3 + c4n + c3(n-1) + c2(n-1) + c5\sum_{j=2}^{n}(k) + c6\sum_{j=2}^{n}(k-1) + c2\sum_{j=2}^{n}(k-1) + c6(n-1)$$

$c5\sum_{j=2}^{n}(k) = c5(\text{n-1})$

$c6\sum_{j=2}^{n}(k-1) = c6(0)$

$$c2\sum_{j=2}^{n}(k-1) = c2(0)$$



$2)\ T(n) = (\text{c4} + \text{c2} + \text{c6} + \text{c5} + \text{c3})\text{n} - (\text{c3} + \text{c2} + \text{c1} + \text{c6})$

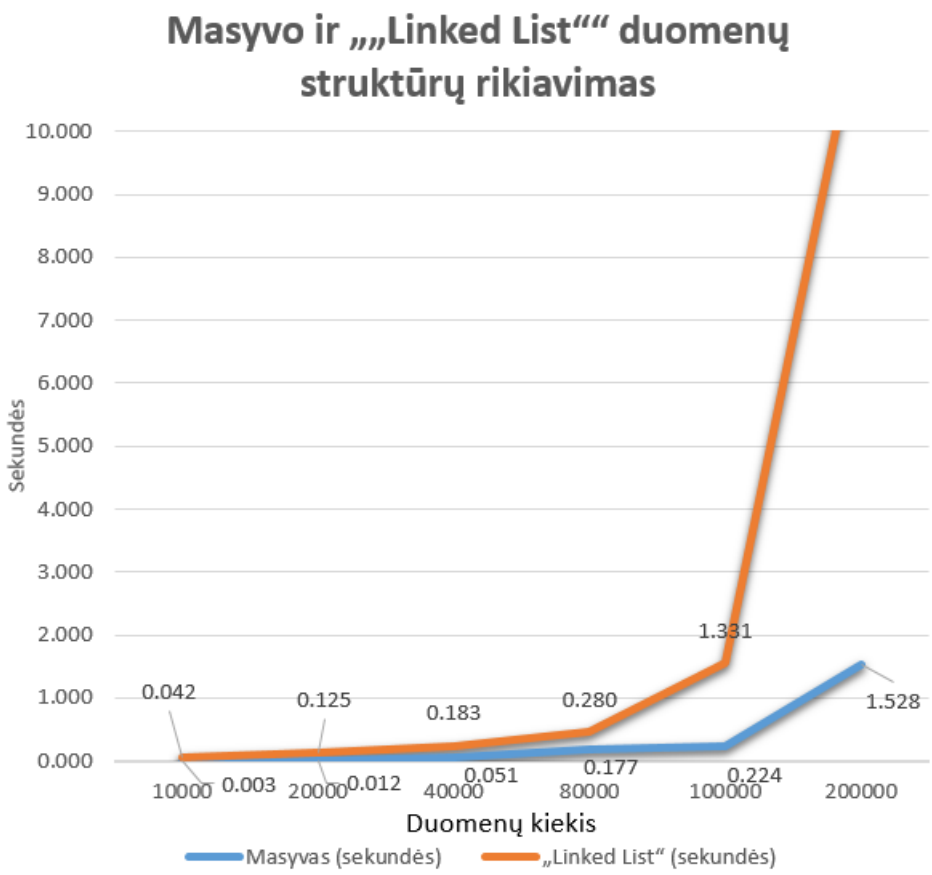
$T(n) = an + b$

Atmetus konstantas gauname, kad InsertionSort metodo sudėtingumas geriausiu atveju yra $\Omega(\boldsymbol{n})$

4. Atlikti eksperimentai

Kai elementai saugomi operatyvioje atmintyje:
Lentelėje pateikti masyvo ir „„Linked List““ duomenų struktūrų rikiavimo laikai sekundėmis priklausomai nuo duomenų kiekio.

Kiekis	Masyvas (sekundės)	„List“ (sekundės)	Masyvas (veiksmai)	„List“ (veiksmai)
10000	0,003	0,042	50781098	24802736
20000	0,012	0,125	199658298	100494888
40000	0,051	0,183	795910942	403385330
80000	0,177	0,428	1105280674	1596119199
100000	0,224	1,331	712861780	1794730627
200000	1,528	10,428	1455217882	1398017097



Pav. 5

Kai elementai saugomi diskinėje atmintyje:

Lentelėje pateikti masyvo ir „„Linked List““ duomenų struktūrų rikiavimo laikai sekundėmis priklausomai nuo duomenų kiekio.

Kiekis	Masyvas (sec)	„List“ (sec)	Masyvas (veiksmai)	„List“ (veiksmai)
500	0,016	-n	122696	-n
1000	0,52	-n	504968	-n
2000	0,128	-n	1971758	-n
4000	0,591	-n	8144212	-n
8000	3,312	-n	32337330	-n
16000	14,238	-n	128293150	-n

Pav. 6

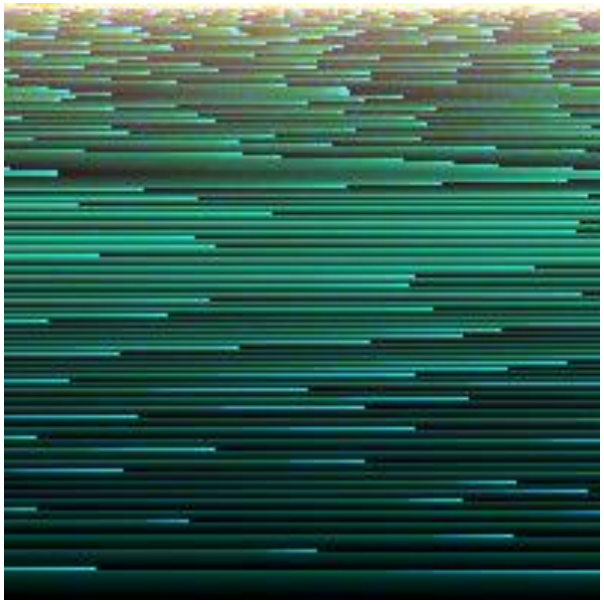
5. Išvados

Insertion sort yra efektyvus algoritmas, kuris rikiuoja mažus elementų skaičius. Insertion sort veikia metodu, kaip dauguma žmonių rikiuoja kortas rankose, žaisdami kortų žaidimus. Mes pradedadame su tuščia kaire ranka ir su kortomis užverstomis ant stalo. Mes pašaliname po viena korta nuo stalo ir įstatome į teisinga poziciją kairėje rankoje. Norint rasti teisinga poziciją, mes lyginame kiekviena korta su esančiomis rankose, nuo dešinės į kairę ir įstatome į tinkama poziciją.

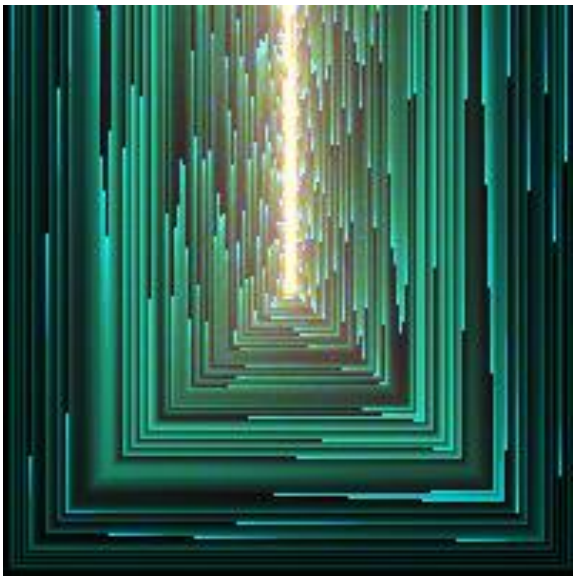
Kiek laiko užtrunka Insertion sort procedūra priklauso nuo įvedamų duomenų kiekio, rikiuojant tūkstantį skaičių užtrunka ilgiau negu rikiuojant tris skaičius. Insertion sort rikiavimas gali užtrukti įvairius laiko tarpus, nes rasta atitinkama vieta gali būti iškarto, arba gali būti pačiame sąrašo gale.

„Insertion Sort“ rikiavimo algoritmas geriausiu atvejais asimptotiškai įvertinamas $\Theta(n)$, o blogiausiu atveju $O(n^2)$.

Surikiuota nuotrauka:



Sudėliota nuotrauka pagal duotą seką:



6. Priedas

Failų parsisiuntimas: https://ufile.io/bceunkt2