

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 2](#_Toc35290788)

[2. Pasirinktos duomenų struktūros realizuotos išorinėje atmintyje struktūrinė diagrama 2](#_Toc35290789)

[Masyvo duomenų struktūros realizavimo išorinėje atmintyje diagrama: 2](#_Toc35290790)

[Sąrašo duomenų struktūros realizavimo išorinėje atmintyje diagrama: 3](#_Toc35290791)

[3. Algoritmo analizė 4](#_Toc35290792)

[Algoritmo įvertinimai literatūroje: 4](#_Toc35290793)

[InsertionSort metodo analizė: 4](#_Toc35290794)

[Blogiausias atvejis: 6](#_Toc35290796)

[Geriausias atvejis: 7](#_Toc35290797)

[4. Atlikti eksperimentai 8](#_Toc35290798)

[Kai elementai saugomi operatyvioje atmintyje: 8](#_Toc35290799)

[Kai elementai saugomi diskinėje atmintyje: 9](#_Toc35290800)

[5. Išvados 10](#_Toc35290801)

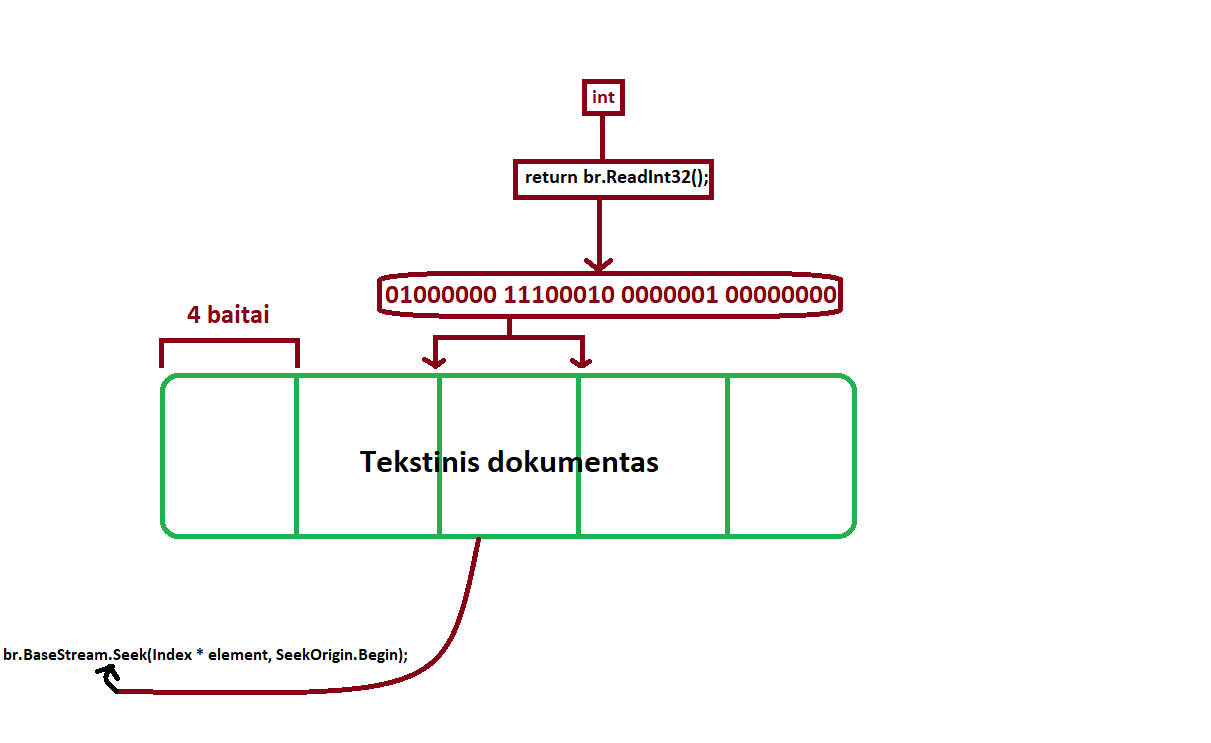
[6. Priedas 10](#_Toc35290802)

# 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.

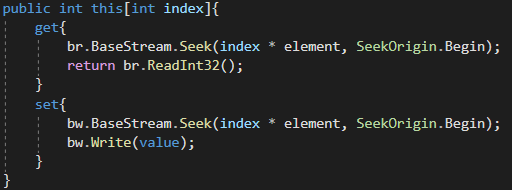
Variantas: 9. Rikiavimas „Insertion Sort“

2. Pasirinktos duomenų struktūros realizuotos išorinėje atmintyje struktūrinė diagrama



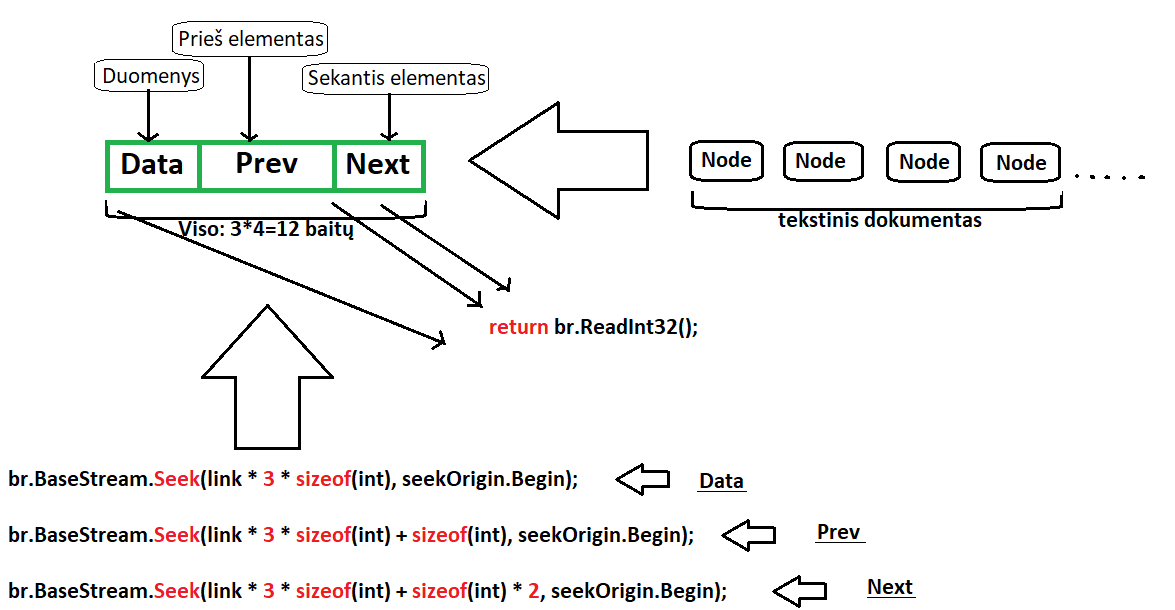
Pav. 1

Programos kodo fragmentas realizuojantis elemento nuskaitymą iš išorinės atminties:



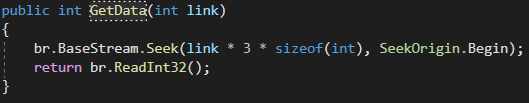
Pav. 2

## Sąrašo duomenų struktūros realizavimo išorinėje atmintyje diagrama:



Pav. 3

Programos kodo fragmentas realizuojantis elemento nuskaitymą iš išorinės atminties:



Pav. 4

# 3. Algoritmo analizė

## Algoritmo įvertinimai literatūroje:

Blogiausiu atveju . Geriausiu atveju

<https://www.bigocheatsheet.com/>

## InsertionSort metodo analizė:

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

**k –surasta tinkama vieta.**

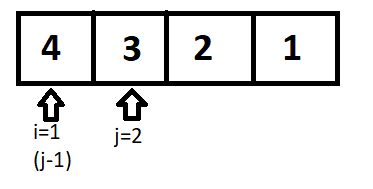
InsertionSort metodo sudėtingumas:

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)

**–** mes turime palyginti kiekvieną elementą. A[j] su visu rikiuotu masyvu A[1...j-1]



)

)

)

+c3(n-1)+c2(n-1)+

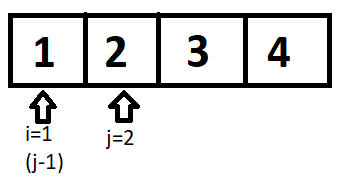
Vertinant asimptotiškai gauname, kad InsertionSort metodo sudėtingumas blogiausiu atveju yra

## **Geriausias atvejis:**

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

n-1)

0)



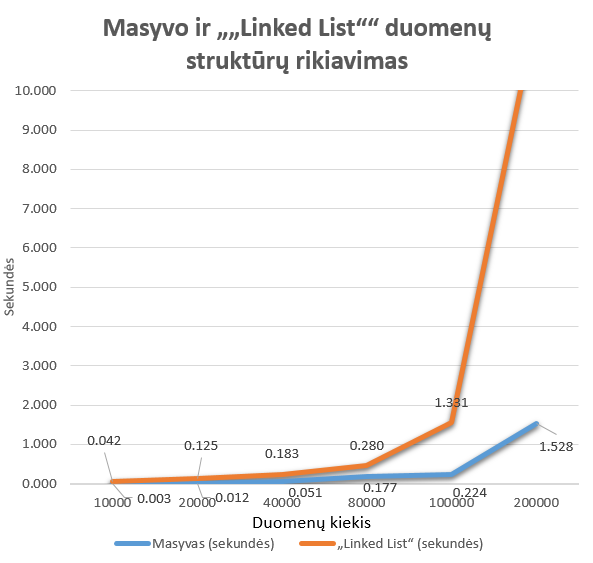
Atmetus konstantas gauname, kad InsertionSort metodo sudėtingumas geriausiu atveju yra

# 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

# 

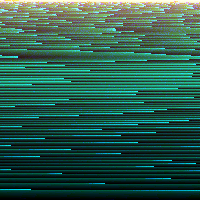
# 5. Išvados

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 tusčia 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.

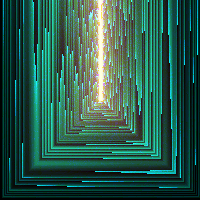
Kiek laiko užtrunka Insertion sort procedūra priklauso nuo įvedamų duomenų kiekio, rikiuojant tūkstantį skaičių užtrunka ilgiau negu rikiuojant tris skaičius. Insertion sort rikiavimas gali užtrukti įvairius laiko tarpus, nes rasta atitinkama vieta gali būti iškarto, arba gali būti pačiame sąrašo gale.

„Insertion Sort“ rikiavimo algoritmas geriausiu atvejais asimptotiškai įvertinamas , o blogiausiu atveju .

**Surikiuota nuotrauka:**



**Sudėliota nuotrauka pagal duotą seką:**



# 6. Priedas

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