Лабораторна робота №1

з дисципліни «Структури даних, аналіз і алгоритми коп'ютерної обробки інформації»

Виконала: студентка групи ІПЗ-11

Манойлова Катерина Борисівна

1. Умова задачі

Написати програму мовою C# з можливістю вибору різних алгоритмів пошуку. Продемонструвати роботу (ефективність, час виконання) програм на різних структурах даних (масив, лінійний зв’язаний список), з різними умовами, що забезпечують зменшення часу виконання. Навести аналіз отриманих результатів. Реалізувати алгоритми:

- пошуку перебором елемента масиву, що дорівнює заданому значенню.
- пошуку з бар'єром елемента масиву, що дорівнює заданому значенню.
- бінарного пошуку елемента масиву рівного заданому значенню.
- бінарного пошуку елемента масиву, рівного заданому значенню, в якій нове значення індексу m визначалося б не як середнє значення між L і R, а згідно з правилом золотого перерізу.

2. Аналіз задачі

Алгоритми пошуку розроблятимемо для двох структур даних: масив та лінійний зв’язаний список. Для зручності структури даних будуть зберігати цілі числа.

Програма буде генерувати масив та заповнювати згенерованими у ньому значеннями лінійний зв’язний список, розмір та діапазон значень вводяться з консолі. Бінарний пошук та його варіація з використанням золотого перетину можливі лише на відсортованих структурах даних. Сортування масиву виконуватимемося за допомогою вбудованого методу класу Array, списку – перетворенням на список відсортованого масиву.

Кожен алгоритм буде виконуватися окремим кейсом конструкції switch, і буде виконуватись для обох структур даних, виводячи результати та час виконання у вигляді таблиці.

3. Структура основних вхідних та вихідних даних

Вхідними даними алгоритмів є відповідна структура даних (масив або лінійний зв’язний список) цілих чисел та шуканий елемент, що є цілим числом.

Вихідними даними є ціле число, що вказує індекс шуканого елементу, або -1 за його відсутності.

Для бінарного пошуку та його варіації з використанням золотого перетину також необхідно вказати коефіцієнт відношення, у якому буде ділитися структура даних при пошуку.

4. Алгоритм розв'язання задачі

1) Лінійний пошук:

Для масиву:

```
i = 0, Pos = 0;
Found = false;

while (i < Array.Length && !Found)
{
   if (Array[i] == SearchVal)
   {
      Pos = i;
      Found = true;
   }
   i++;
}
if (Found)
   return Pos;
else
   return -1;
```

Для списку:

```
i = 0, Pos = 0;
Found = false;

Node<int> Val = list.head;

while (Val != null && !Found)
{
   if (Val.Data == SearchVal)
   {
      Pos = i;
      Found = true;
   }
   i++;
   Val = Val.Next;
}

if (Found)
   return Pos;
else
   return -1;
```

2) Пошук з бар’єром

Для масиву:

```
i = 0;

int[] tempArr = new int[Array.Length + 1];
```

```
while (i < Array.Length)
{
   tempArr[i] = Array[i];
   i++;
}

tempArr[i] = SearchVal;
i = 0;

while (tempArr[i] != SearchVal)
{
   i++;
}

if (i == Array.Length)
   return -1;
else
   return i;
```

Для списку:

```
i = 0;
LinkedList<int> newlist = new LinkedList<int>();
Node<int> N = list.head;
for (int j = 0; j < list.Count; j++)
{
   newlist.Add(N.Data);
   N = N.Next;
}
newlist.Add(SearchVal);

Node<int> Val = newlist.head;

while (Val.Data != SearchVal)
{
   i++;
   Val = Val.Next;
}

if (i >= list.Count)
   return -1;
else
   return i;
```

3) Бінарний пошук та його варіація з золотим перетином:

Для масиву:

```
Left = 0, Right = Array.Length - 1;
```

```
while (Left <= Right)
{
   int Mid = (Left + (Right - Left) / Divider);

   if (Array[Mid] == SearchVal)
      return Mid;

   if (Array[Mid] < SearchVal)
      Left = Mid + 1;
   else
      Right = Mid - 1;
}

return -1;
```

Для списку:

```
Left = 0, Right = list.Count - 1;
while (Left <= Right)
{
   int Mid = (Left + (Right - Left) / Divider);

   if (FindEl(list, Mid).Data == SearchVal)
      return Mid;

   if (FindEl(list, Mid).Data < SearchVal)
      Left = Mid + 1;
   else
      Right = Mid - 1;
}

return -1;
```

Алгоритм пошуку індексу елемента списку:

```
Node<int> node = list.First;
for (int i = 0; i < Index; i++)
{
   node = node.Next;
}
return node;
```

5. Текст програми

Текст програми у репозиторії GitHub за посиланням:
https://github.com/ManoilovaKaterina/Lab_Report/tree/main/%D0%90%D0%A1%D0%94/%D0%90%D0%A1%D0%94%20%D0%BB1

6. Набір тестів

Для кожного алгоритму пошуку вирішено зробити по два тести на структурах даних різних розмірів з двома контрольними значеннями: наявний та відсутній у структурах даних елемент.

Тест 1:

10 елементів:

57 86 16 95 31 86 79 92 49 78

Контрольні значення: 92, 100

Тест 2:

1000 елементів:

5584 4107 702 9524 5011 4707 5498 3066 7864 4248 3481 2526 382 8077 7102 4205 2430 7006 900 7227 6311 4300 5383 8387 1559 4067 4180 5845 2582 8820 3373 6468 1951 7681 3310 9250 8499 8762 7799 9536 5289 1241 6439 7568 5437 5758 5915 5844 3182 4862 2917 2788 6918 6516 7367 1284 8723 2315 7965 944 527 9652 483 9043 874 7013 574 2700 4767 7851 5705 3667 9207 1363 1938 5070 7860 7815 2949 5800 8152 8336 2662 7719 5902 585 9549 5434 314 2026 527 6184 797 6855 9009 5636 757 7396 6693 8423 5183 3214 1077 5331 9156 9249 3580 5554 4578 2296 3423 907 9365 2164 2792 2190 6990 2763 3140 288 7464 5139 2385 2740 7324 9521 2869 2351 2354 6301 4312 464 1121 4525 617 4937 7259 7331 8563 6652 7005 3994 856 8017 8602 9619 6819 8633 4062 6818 8646 7993 4255 6405 959 43 828 8336 8007 9635 6380 1228 3200 8276 7984 2959 9786 4839 1547 7855 4931 9659 4200 6487 3283 3469 4283 4368 4138 7704 2702 4236 8289 5535 7655 6318 6208 4716 3566 573 4108 8923 9323 8927 272 5776 794 2579 33 5643 9833 1979 7086 6207 1887 8987 3792 7767 3121 7490 5760 6460 4198 302 6933 2143 2939 7664 621 1665 6750 5069 1273 973 3746 6007 335 5273 6214 7507 2675 1703 4334 8494 7871 722 7149 2081 3648 8668 2525 8441 1594 6076 4813 7647 4725 9021 1994 8129 2836 3129 1958 8368 8893 4763 706 6113 2460 5879 8651 8519 1552 5614 4814 4057 2126 5703 2430 6210 4993 858 4016 1952 9139 8308 3474 5197 6160 6099 1282 1313 3278 8085 4671 9545 9744 5966 9601 3107 16 1036 9621 7768 16 4006 6888 5980 1261 756 5521 9021 6590 5783 3135 1978 9113 5 9228 584 1288 5508 9952 6361 4596 7338 5240 3466 943 5269 4312 6159 6102 9322 6193 3956 1237 6248 1936 5132 1420 7494 3936 5404 577 2261 4722 7495 4950 2693 432 9738 6737 9016 1818 4507 1084 3259 3171 2678 8765 3421 5036 5992 6443 9361 2918 7267 9589 9178 740 2865 8069 4095 9164 3794 1572 4548 5783 2335 2615 7745 8516 8250 4836 4573 9422 7086 7504 1686 2872 7977 3076 9258 6367 7998 2458 7943 8960 1216 9342 7454 7905 5772 1952 7567 1669 2641 9852 8024 2935 6535 7476 836 62 1020 4905 6786 1155 1869 9938 5832 9763 7903 6305 2763 9788 8810 7728 1165 1336 3628 5587 4566 2992 5160 9840 2744 6297 7269 2904 6780 7233 9479 8751 6336 501 2240 9195 5346 3092 5671 6788 7091 1277 3510 7691 1 9466 9188 7778 2858 4912 8686 6688 9306 948 2909 7843

4902 1180 2161 488 3886 8964 3158 8598 284 9152 9968 2262 7548 9615 2382 8073 5664 6840 8496 3289 9482 7469 9838 3278 7109 9490 46 1868 8547 2790 9445 5388 7591 4397 4293 4166 931 5182 2901 8126 8119 4911 7406 849 9497 6926 230 3243 2530 613 1024 2465 2451 9619 8360 7432 1342 8883 3379 4395 8918 1289 51 7493 9707 4580 4670 3151 5321 8215 7141 481 3938 370 5170 4571 62 8988 3780 182 9260 6803 9337 7934 1766 6979 2937 7971 6036 6861 2823 2048 1803 8506 9208 6829 4860 9912 1142 4917 2255 7079 7921 4314 3471 542 8526 2081 4449 3789 7369 2353 5102 3196 5134 2114 1952 2116 5727 2727 1642 6698 7114 8460 5392 5093 8678 5432 1161 8340 9710 150 7846 8863 7927 2181 8881 5023 4463 1223 8453 855 3522 2246 9491 470 6945 8607 3371 7093 4877 2743 4185 8414 3274 5556 9964 9461 8922 8377 1864 3094 919 6109 4078 7219 4506 6238 5269 2953 3232 6928 7653 4503 4274 787 3176 4867 8969 4921 8148 71 2060 4068 3462 6967 5965 9432 5589 2370 2216 9420 6100 6085 9359 5359 9936 7412 8167 2271 5964 706 3338 418 3861 7948 5089 9681 4140 2487 2380 5096 492 4001 229 1793 1033 6851 2646 7111 1254 5074 649 2898 736 3812 828 1568 5144 8915 850 5763 6700 6697 8957 7441 6732 1641 207 5514 1877 6283 5292 3102 9990 7120 8928 2099 5856 7566 4325 3084 4766 1056 1016 889 57 439 9682 48 7120 3521 4537 5225 1636 6616 8396 3794 5043 2788 5060 9391 6644 7132 5233 4971 9781 7546 2908 3616 4884 8728 5712 7577 4589 7766 997 5643 5681 8067 7383 6293 1959 158 8394 8356 1746 67 1466 3373 8723 5133 7055 3068 2506 4934 6439 7634 5822 6045 1107 2511 7531 6066 5164 8391 7809 6959 635 3870 5619 2752 8112 6976 5404 8766 7432 6486 8738 6877 3224 9714 6080 9534 4892 9750 1673 2643 5071 9655 1326 3362 9821 9636 6959 4872 8761 5893 4993 2 546 4786 9431 7596 7754 5971 7021 78 7663 3739 7501 9953 8896 8944 2820 1393 6430 7997 1173 5414 6717 5165 1887 980 2543 2256 2931 8476 17 532 5 1538 1492 8735 6331 8437 283 8484 1779 8921 2729 1594 4980 1332 2153 1372 4466 877 6815 5566 8441 764 4719 9579 3284 5381 2899 8451 5052 5498 3040 8544 61 7131 3733 5963 8461 161 2612 4383 1110 7945 6218 2252 2685 5123 185 555 8827 1170 7789 2053 1661 4451 2090 9240 6819 1912 5450 950 5538 1832 3432 6280 5881 4184 1533 7848 7598 6190 2911 4305 8264 2431 4456 495 4546 2466 6894 8161 5195 2343 9624 3881 7709 986 6882 5610 5041 4493 9143 6548 4710 1662 8844 9278 4513 9937 6371 8500 3590 2336 2956 2637 8259 3483 3789 9229 9995 1594 4694 4353 5017 7288 5754 3194 2208 9550 8571 4894 7301 5922 2807 3105 7046 6362 9595 6601 3586 5177 5982 4608 6094 8394 4571 2859 9386 6986 5622 4225 7196 7653 5615 3446 9798 4790 1530 7421 5908 5649 7070 4963 1366 1477 1198 7667 9528 9851 5590 1896 3888

Контрольні значення: 1, 10000

7. Результати тестування програми та аналіз отриманих помилок

Лінійний пошук:

Тест 1:

```
--------------------Лінійний пошук--------------------

Введіть шукане значення: 92
Значення "92" знайдене в даному масиві за індексом 7
Значення "92" знайдене в даному списку за індексом 7

====================Час виконання====================
Для масиву:     00:00:00.0000025
Для списку:     00:00:00.0000040
=====================================================

Оберіть команду: 2

--------------------Лінійний пошук--------------------

Введіть шукане значення: 100
Значення "100" не знайдене в масиві.
Значення "100" не знайдене у списку.

====================Час виконання====================
Для масиву:     00:00:00.0000035
Для списку:     00:00:00.0000057
=====================================================
```

Тест 2:

```
--------------------Лінійний пошук--------------------

Введіть шукане значення: 1
Значення "1" знайдене в даному масиві за індексом 447
Значення "1" знайдене в даному списку за індексом 447

====================Час виконання====================
Для масиву:     00:00:00.0000064
Для списку:     00:00:00.0000098
=====================================================

Оберіть команду: 2

--------------------Лінійний пошук--------------------

Введіть шукане значення: 10000
Значення "10000" не знайдене в масиві.
Значення "10000" не знайдене у списку.

====================Час виконання====================
Для масиву:     00:00:00.0000112
Для списку:     00:00:00.0000164
=====================================================
```

Пошук з бар’єром:

Тест 1:

```
--------------------Пошук з бар'єром--------------------

Введіть шукане значення: 92
Значення "92" знайдене в даному масиві за індексом 7
Значення "92" знайдене в даному списку за індексом 7

====================Час виконання====================
Для масиву:     00:00:00.0000037
Для списку:     00:00:00.0000054
=====================================================

Оберіть команду: 3

--------------------Пошук з бар'єром--------------------

Введіть шукане значення: 100
Значення "100" не знайдене в масиві.
Значення "100" не знайдене у списку.

====================Час виконання====================
Для масиву:     00:00:00.0000042
Для списку:     00:00:00.0000072
=====================================================
```

Тест 2:

```
--------------------Пошук з бар'єром--------------------

Введіть шукане значення: 1
Значення "1" знайдене в даному масиві за індексом 447
Значення "1" знайдене в даному списку за індексом 447

====================Час виконання====================
Для масиву:     00:00:00.0000269
Для списку:     00:00:00.0001171
=====================================================

Оберіть команду: 3

--------------------Пошук з бар'єром--------------------

Введіть шукане значення: 10000
Значення "10000" не знайдене в масиві.
Значення "10000" не знайдене у списку.

====================Час виконання====================
Для масиву:     00:00:00.0000343
Для списку:     00:00:00.0001162
=====================================================
```

Бінарний пошук:

Тест 1:

```
--------------------Бінарний пошук--------------------

Введіть шукане значення: 92
Значення "92" знайдене в даному масиві за індексом 8
Значення "92" знайдене в даному списку за індексом 8

====================Час виконання====================
Для масиву:     00:00:00.0000011
Для списку:     00:00:00.0000034
=====================================================

Оберіть команду: 4

--------------------Бінарний пошук--------------------

Введіть шукане значення: 100
Значення "100" не знайдене в масиві.
Значення "100" не знайдене у списку.

====================Час виконання====================
Для масиву:     00:00:00.0000007
Для списку:     00:00:00.0000016
=====================================================
```

Тест 2:

```
--------------------Бінарний пошук--------------------

Введіть шукане значення: 1
Значення "1" знайдене в даному масиві за індексом 0
Значення "1" знайдене в даному списку за індексом 0

====================Час виконання====================
Для масиву:     00:00:00.0000046
Для списку:     00:00:00.0000226
=====================================================

Оберіть команду: 4

--------------------Бінарний пошук--------------------

Введіть шукане значення: 10000
Значення "10000" не знайдене в масиві.
Значення "10000" не знайдене у списку.

====================Час виконання====================
Для масиву:     00:00:00.0000079
Для списку:     00:00:00.0001167
=====================================================
```

Бінарний пошук за золотим перетином:

Тест 1:

```
--------------------Бінарний пошук за золотим перетином-

Введіть шукане значення: 92
Значення "92" знайдене в даному масиві за індексом 8
Значення "92" знайдене в даному списку за індексом 8

====================Час виконання====================
Для масиву:     00:00:00.0000013
Для списку:     00:00:00.0000018
=====================================================

Оберіть команду: 5

--------------------Бінарний пошук за золотим перетином-

Введіть шукане значення: 100
Значення "100" не знайдене в масиві.
Значення "100" не знайдене у списку.

====================Час виконання====================
Для масиву:     00:00:00.0000014
Для списку:     00:00:00.0000027
=====================================================
```

Тест 2:

```
--------------------Бінарний пошук за золотим перетином-

Введіть шукане значення: 1
Значення "1" знайдене в даному масиві за індексом 0
Значення "1" знайдене в даному списку за індексом 0

====================Час виконання====================
Для масиву:     00:00:00.0000020
Для списку:     00:00:00.0000325
=====================================================

Оберіть команду: 5

--------------------Бінарний пошук за золотим перетином-

Введіть шукане значення: 10000
Значення "10000" не знайдене в масиві.
Значення "10000" не знайдене у списку.

====================Час виконання====================
Для масиву:     00:00:00.0000022
Для списку:     00:00:00.0001444
=====================================================
```

Висновок:

Лінійний зв’язний список: лінійний пошук для списку та для масиву мають близький час виконання, за ним не спостерігається значних стрибків у часі в залежності від розміру структури. Пошук з бар'єром у списку невеликого розміру займає приблизно стільки ж часу, скільки й лінійний, однак на великих розмірах він має найбільший час виконання. Бінарний пошук та його модифікація за золотим перетином мають найменший час виконання на списку маленького розміру, але витрачають багато часу на великих розмірах.

Отже, лінійний пошук є найбільш ефективним для списку, пошук з бар’єром – найменш ефективним. Бінарні пошуки мають незначну перевагу лише на невеликих розмірах.

Масив: пошук з бар’єром займає дещо більше часу, ніж лінійний. Різниця є більш помітною на великих розмірах. Бінарний пошук є найбільш ефективним для масиву, маючи найменший час виконання з усіх алгоритмів. Різниця між звичайним бінарним пошуком та модифікованим за золотим перетином є незначною, більш того – розмір масиву майже не впливає на час виконання алгоритму.

8. Аналіз помилок, допущених в ході розробки

1) Розробка алгоритму пошуку з бар’єром.

Для масиву:

Оскільки шуканий елемент і є «бар’єром», шукане значення рано чи пізно буде досягнуто, тому треба визначити показник ненаявності елементу в початковому масиві. Першим варіантом вирішення цього моменту було додавання нової змінної *j*.

```
static void ArrBarSearch(int[] Array, int SearchVal)
{
    int i = 0;
    int[] tempArr = new int[Array.Length + 1];

    while (i < Array.Length)
    {
        tempArr[i] = Array[i];
        i++;
    }
    tempArr[i] = SearchVal;

    int j = 0;
    while (tempArr[j] != SearchVal)
    {
        j++;
    }

    if (j==i)
        Console.WriteLine("Елемент " + SearchVal + " не знайдено в даному масиві.");
    else
        Console.WriteLine("Елемент " + SearchVal + " знайдено за індексом " + j);
}
```

Порівняння *j* з *i* є показником наявності елементу в массиві.

Однак, введення нової змінної не є необхідним. Максимальний індекс масиву завжди на 1 менше його величини, отже як тільки *i* досягає значення величини – відбувається вихід за межі початкового масиву.

```
static void ArrBarSearch(int[] Array, int SearchVal)
{
    int i = 0;
    int[] tempArr = new int[Array.Length + 1];

    while (i < Array.Length)
    {
        tempArr[i] = Array[i];
        i++;
    }

    tempArr[i] = SearchVal;
    i = 0;

    while (tempArr[i] != SearchVal)
    {
        i++;
    }

    if (i == Array.Length)
        Console.WriteLine("Елемент " + SearchVal + " не знайдено в даному масиві.");
    else
        Console.WriteLine("Елемент " + SearchVal + " знайдено за індексом " + i);
}
```

2) Розробка алгоритму бінарного пошуку та його варіації

Для масиву:

```
static int ArrGoldBinSearch(int[] Array, int SearchVal)
{
    int Left = 0, Right = Array.Length - 1;
    while (Left <= Right)
    {
        int Gold = (int)(Left + (Right - Left) / ((Math.Sqrt(5) + 1) / 2));

        if (Array[Gold] == SearchVal)
        {
            return Gold;
        }

        if (Array[Gold] < SearchVal)
        {
            Left = Gold + 1;
        }
        else
        {
            Right = Gold - 1;
        }
    }

    return -1;
}
```

Бінарний пошук за золотим перетином відрізняється від звичайного дільником: замість ділення навпіл, відбувається ділення на число фі.

З технічної точки зору, даний вид пошуку може працювати з будь-яким дільником, не менше 1. Ділити масив можна у будь-якому відношенні, основна думка залишається незмінною: скорочення інтервалів пошуку елементу.

Отже, у фінальній версії було вирішено об'єднати бінарний пошук з його варіацією з золотим перетином, додавши параметр Divider, який визначає відношення.

Для списку:

Оскільки у зв’язному списку неможливо дістатися до елементів за індексом, застосування алгоритму бінарного пошуку до цієї структури даних спочатку здалося надскладним. Першою ідеєю було «обрізати» непотрібну частину елементів при скороченні розглядаємої частини списку.

```
static int ListBinSearch(LinkedList<int> list, int SearchVal)
{
    int i = 0;
    while (list.Count>0)
    {
        LinkedListNode<int> Mid = FindMiddle(list);

        if (Mid.Value == SearchVal)
        {
            return i;
        }

        if (Mid.Value < SearchVal)
        {
            while(list.First.Value != Mid.Value)
            {
                list.RemoveFirst();
                i++;
            }
        }
        else
        {
            while (list.Last.Value != Mid.Value)
            {
                list.RemoveLast();
                i--;
            }
        }
    }

    return -1;
}
```

```
Згенерований список:
1 4 2 7 7 0 7 3 5 0
Оберіть команду:
9
0 0 1 2 3 4 5 7 7 7
0
Значення "0" знайдене в даній структурі даних за індексом -7
Оберіть команду:
9
0 0 1 2 3 4 5 7 7 7
1
Значення "1" знайдене в даній структурі даних за індексом -6
Оберіть команду:
9
0 0 1 2 3 4 5 7 7 7
7
Значення "7" знайдене в даній структурі даних за індексом 5
Оберіть команду:
9
0 0 1 2 3 4 5 7 7 7
100
```

Дана ідея виявилась абсолютно неефективною. Окрім псування початкового списку, було необхідно придумати спосіб підрахунку індексів, а також нову умову закінчення циклу. Загалом, навіть якщо цей метод довести до вірно працюючого варіанту, він буде занадто сильно відрізнятися від алгоритму бінарного пошуку, залишаючи лише спільний задум у скороченні інтервалу пошуку.

Було вирішено повернутися до алгоритму бінарного пошуку у масиві, знайшовши індекс елементу списку за допомогою перетворення списку до масиву та використання методу класу Array.

```
static int FindInd(LinkedList<int> list, int FoundVal)
{
    var Arr = list.ToArray();
    var i = Array.IndexOf(Arr, FoundVal);
    return i;
}
```

```
static int ListBinSearch(LinkedList<int> list, int SearchVal)
{
    while (list.Count>0)
    {
        LinkedListNode<int> Mid = FindMiddle(list);

        if (Mid.Value == SearchVal)
        {
            return FindInd(list, SearchVal);
        }

        if (Mid.Value < SearchVal)
        {
            while(list.First.Value != Mid.Value)
            {
                list.RemoveFirst();
            }
        }
        else
        {
            while (list.Last.Value != Mid.Value)
            {
                list.RemoveLast();
            }
        }
    }

    return -1;
}
```

```
Згенерований список:
2 3 9 6 2 5 1 7 2 0
Оберіть команду:
9
0 1 2 2 2 3 5 6 7 9
2
Значення "2" знайдене в даній структурі даних за індексом 2
Оберіть команду:
9
0 1 2 2 2 3 5 6 7 9
3
Значення "3" знайдене в даній структурі даних за індексом 5
```

Однак, проблема в умові закінчення циклу все ще не є вирішеною. Проте на даному етапі з’явилася інша думка: модифікувати функцію знаходження

середнього елементу для знаходження елемента за будь-яким індексом, таким чином повернувшись до початкової логіки бінарного пошуку для масиву.

Функція знаходження елементу списку за індексом:

```
static LinkedListNode<int> FindEl(LinkedList<int> list, int Index)
{
    LinkedListNode<int> Node = list.First;
    for (int i = 0; i < Index; i++)
    {
        Node = Node.Next;
    }
    return Node;
}
```

Таким чином, алгоритм бінарного пошуку для списку залишається майже ідентичним пошуку у масиві.

```
static int ListBinSearch(LinkedList<int> list, int SearchVal)
{
    int Left = 0, Right = list.Count - 1;
    while (Left <= Right)
    {
        int Mid = Left + (Right - Left) / 2;

        if (FindEl(list, Mid).Value == SearchVal)
        {
            return Mid;
        }

        if (FindEl(list, Mid).Value < SearchVal)
        {
            Left = Mid + 1;
        }
        else
        {
            Right = Mid - 1;
        }
    }

    return -1;
}
```