

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №5
по курсу «Алгоритмы и структуры данных»
Тема: Деревья. Пирамида, пирамидальная сортировка.
Очередь с приоритетами.
Вариант 11

Выполнил:
Тимаков Е.П. (фамилия имя)
К3141 (номер группы)

Проверила:
Артамонова В.Е.

Санкт-Петербург 2024 г.

Содержание отчета

Содержание отчета	2
Задачи по варианту	2
Задача №2. Высота дерева	2
Задача №7. Снова сортировка	5
Вывод	9

Задачи по варианту

Задача №2. Высота дерева

В этой задаче ваша цель - привыкнуть к деревьям. Вам нужно будет прочитать описание дерева из входных данных, реализовать структуру данных, сохранить дерево и вычислить его высоту.

- Вам дается корневое дерево. Ваша задача - вычислить и вывести его высоту. Напомним, что высота (корневого) дерева - это максимальная глубина узла или максимальное расстояние от листа до корня. Вам дано произвольное дерево, не обязательно бинарное дерево.

- Формат ввода или входного файла (input.txt). Первая строка содержит число узлов n ($1 \leq n \leq 105$). Вторая строка содержит n целых чисел от -1 до $n-1$ – указание на родительский узел. Если i -ое значение равно -1 , значит, что узел i - корневой, иначе это число является обозначением индекса родительского узла этого i -го узла ($0 \leq i \leq n - 1$). Индексы считать с 0. Гарантируется, что дан только один корневой узел, и что входные данные представляют дерево.

- Формат вывода или выходного файла (output.txt). Выведите целое число – высоту данного дерева.

- Ограничение по времени. 3 сек.

- Ограничение по памяти. 512 мб.

Листинг кода. (именно листинг, а не скрины)

```
def main():
    with open("input.txt") as f:
        n = int(f.readline())
        nums = list(map(int, f.readline().split()))
    x = max(nums)
    k = 1
    while x != -1:
        x = nums[x]
        k += 1
    with open("output.txt", "w", encoding="utf-8") as f:
        f.write(str(k))
```

main()

Текстовое объяснение решения:

Первая строка файла содержит число "n", а вторая строка содержит список чисел. Затем код находит максимальное значение в списке и использует его в качестве индекса для доступа к другому элементу в списке. Этот процесс продолжается, пока не будет найден элемент со значением -1.

Результат работы кода на примерах из текста задачи:

input.txt:

1	5
2	4 -1 4 1 1

output.txt:

1	3
---	---

Результат работы кода на максимальных:

input.txt:

1	10000
2	-1 0 0 2 0 2 3 1 3 2 6 0 10 3 13 4 12 16 17 12 1 17 19 2 23 18 13 16 1 1 15 16 12 18 1 6 1 19 10 36 22 19 37 18 22 9 33 9 17 38 14 30 34 15 7 8 30 22 34 53 44 20 52 51 25 30 52 4 22 67 33 53 62 42 23 39 54 1 77 15 26 11 8 56 22 34 55 52 82 16 26 16 37 56 36 35 73 23 0 31 63 27 52 2 22 97 16 90 10 40 54 42 87 13 69 28 85 22 51 96 19 101 36 11 44 114 29 108 93 90 70 115 48 124 52 28 103 86 78 57 83 117 63 5 64 128 117 41 145 27 93 80 67 35 66 36 102 60 81 25 55 90 113 59 78 144 20 46 149 72 91 122 76 166 45 147 139 8 88 116 151 109 47 50 74 159 121 144 3 138 75 27 189 5 166 57 136 76 20 35 43 172 17 84 69 23 53 201 192 1 63 32 49 105 31 31 57 107 186 137 112 35 12 20 181 112 90 135 1 68 92 60 1 157 222 162 144 231 185 40 70 108 195 115 41 84 109 130 71 65 220 235 148 42 164 233 97 199 73 141 12 1 10 27 41 230 62 108 199 125 101 102 170 154 18 269 123 99 183 253 135 256 280 268 170 278 275 89 266 244 235 200 178 70 241 266 8 118 190 93 163 290 24 40 179 21 174 294 72 196 296 32 240 287 212 13 267 75 138 263 315 192 273 101 309 78 247 175 209 68 295 124 92 132 280 224 212 43 83 205 218 288 227 33 28 334 171 238 124 317 312 157 7 205 28 38 276 343 17 330 226 153 136 93 284 235 207 267 78 333 153 210 132 66 22 81 291 296 227 327 108 130 348 23 30 261 200 367 125 327 312 362 163 317 378 276 389 52 221 212 188 194 383 375 299 391 230 139 150 215 323 24 100 202 10 255 371 217 47 74 191 213 307 82 129 99 424 402 255 196 338 95 419 211 105 59 54 408 285 5 52 361 411 68 176 321 395 91 426 269 29 1 326 93 431 419 430 233 388 445 133 102 403 150 258 405 70 309 440 74 228 224 471 137 259 258 327 66 265 205 8 442 384 125 96 412 482 470 393 370 262 302 355 154 190 188 412 199 54 195 2 78 218 348 449 438 143 226 322 397 439 307 10 276 444 258 236 257 418 403 71 350 152 192 470 249 54 204 396 367 240 299 357 234 408 32 216 310 192 251 233 434 98 267 184 484 91 47 498 403 478 257 428 466 514 492 217 262 465 422 498 145 4 502 380 294 443 436 491 437 461 328 226 70 374 119 197 48 370 215 391 562 457 8 31 87 91 79

output.txt:

1	12
---	----

Результат работы кода на минимальных значениях:

input.txt:

1	1
2	-1

output.txt:

1	1
---	---

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.001	25
Пример из задачи	0.002	25
Верхняя граница диапазона значений входных данных из текста задачи	0.019	30

Вывод по задаче:

В данной задаче, мы ознакомились с структурой дерева. Также мы реализовали алгоритм подсчета высоты дерева.

Задача №7. Снова сортировка

Напишите программу пирамидальной сортировки на Python для последовательности в убывающем порядке. Проверьте ее, создав несколько случайных массивов, подходящих под параметры:

- Формат входного файла (input.txt). В первой строке входного файла содержится число n ($1 \leq n \leq 105$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 109.

- Формат выходного файла (output.txt). Одна строка выходного файла с отсортированным по невозрастанию массивом. Между любыми двумя числами должен стоять ровно один пробел.

- Для проверки можно выбрать случай, когда сортируется массив размера 103, 104, 105 чисел порядка 109, отсортированных в обратном порядке; когда массив уже отсортирован в нужном порядке; когда много одинаковых элементов, всего 4-5 уникальных; средний - случайный. Сравните на данных сетах Randomized-QuickSort, MergeSort, HeapSort, InsertionSort.

Листинг кода. (именно листинг, а не скрины)

```
class Heap:
    def __init__(self, array=None):
        if array is None:
```

```

        array = []
    self.array = array
    self.length = len(array)
    self.heapify_full()

def add(self, value):
    self.array.append(value)
    self.length += 1
    if self.length > 1:
        self.heapify_full()

def pop_minimal(self):
    if self.length > 0:
        value = self.array.pop(0)
        self.length -= 1
        self.heapify_full()
        return value
    return None

def heapify(self, arr, n, i):
    left_ch = i * 2 + 1
    right_ch = i * 2 + 2
    smallest = i
    if left_ch < n and arr[smallest] > arr[left_ch]:
        smallest = left_ch
    if right_ch < n and arr[smallest] > arr[right_ch]:
        smallest = right_ch
    if smallest != i:
        arr[i], arr[smallest] = arr[smallest], arr[i]
        self.heapify(arr, n, smallest)

def heapify_full(self):
    for i in range(self.length - 1, -1, -1):
        self.heapify(self.array, self.length, i)

def heap_sort(self):
    arr = self.array[:]
    for i in range(self.length - 1, 0, -1):
        (arr[i], arr[0]) = (arr[0], arr[i])

```

```

        self.heapify(arr, i, 0)
    return arr

def change(self, value_to_change, value_to_change_to):
    if value_to_change in self.array:
        to_change = self.array.index(value_to_change)
        self.array[to_change] = value_to_change_to
        self.heapify_full()

def main():
    with open("input.txt") as f:
        _ = f.readline()
        nums = list(map(int, f.readline().split()))
        heap = Heap(nums)
        sorted_nums = heap.heap_sort()

    with open("output.txt", "w", encoding="utf-8") as f:
        f.write(" ".join(map(str, sorted_nums)))

main()

```

Текстовое объяснение решения:

Данный код реализует класс Heap (куча), который представляет структуру данных для хранения элементов с возможностью эффективного извлечения наименьшего элемента и сортировки. В классе реализованы методы для инициализации кучи, добавления элемента, извлечения минимального элемента, сортировки кучи, изменения значения элемента в куче.

Функция main читает входные данные из файла "input.txt", создает объект класса Heap, сортирует числа с помощью метода heap_sort.

Результат работы кода на примерах из текста задачи:

input.txt:

1	5
2	4 -1 4 1 1

output.txt:

1	β

Результат работы кода на максимальных:

input.txt:

1	1000000
2	377083 89333 233497 809575 707927 211704 -575854 -848919 368762 178247 -753174 -738753 -993382 543717 -115216 294209 673820 834845 -660655 -336902 99877 -452545 -840187 -313655 -595479 -812467 230627 -360135 282972 -428158 182442 -525500 -356873 324252 -928463 -209015 151031 -440979 -853635 793503 -984773 450870 39528 -571168 818868 -558371 -239428 583542 -834590 706162 -14346 506820 450085 -474501 -121392 712471 569568 319735 437113 -891365 -547178 354832 -659722 -299606 -763309 -6602 59913 -126584 909019 729796 -741292 -955990 -90538 -270500 -461647 -561586 74852 -617278 144476 -884573 -564329 173814 733052 -764720 948908 17163 313295 -286836 321959 288283 -99089 -587025 733506 696022 552919 -745774 -145004 74094 800871 -605263 914286 -569782 648005 280941 -248769 876131 -816458 118630 -656509 -736627 -766864 -707634 -972516 -251285 -431579 -226604 425529 -155797 10019 993297 363053 524881 -138609 290797 406418 234328 693422 33052 443560 481986 224320 -715926 226021 -642142 562352 -239041 -713460 -224421 -437584 755776 80668 388523 -369658 665043 333841 727270 412852 -543668 -721325 -731762 167461 -439708 -336837 -276876 -705460 -177860 339016 -967991 -666798 267001 -665412 771322 -480240 308636 -154830 -697602 379618 -715214 172957 160260 -708922 619506 237330 -766198 -795123 -722990 -460920 566773 -806547 915554 902641 60200 870201 441778 151093 498777 430480 -202931 122050 34893 -370866 193727 -791163 860395 -580562 -889567 334462 289735 -235714 455427 -814052 -585487 972235 318341 113280 -727747 266973 -653577 386444 -535421 612690 566421 673031 443757 -86593 -374248 -485765 -556600 -134253 -880491 288364 605925 231937 727235 -376753 123367 -795173 -719404 981479 859275 -52320 -840844 -926560 -564871 195059 325109 -465386 649534 74309 -514116 349474 -905335 964902 -71485 -37273 103326 -615514 497456 550223 -955948 201641 346549 337088

output.txt:

1	999997 999996 999994 999994 999993 999988 999986 999986 999985 999985 999985 999985 999984 999983 999982 999982 999981 999979 >
	999977 999975 999972 999971 999970 999967 999966 999965 999961 999961 999957 999954 999953 999953 999952 999951 999949 999948 >
	999941 999940 999939 999939 999937 999934 999933 999932 999930 999929 999926 999921 999917 999916 999916 999915 999915 999911 >
	999911 999910 999910 999908 999906 999904 999903 999903 999900 999900 999895 999895 999894 999891 999888 999887 999886 999883 >
	999878 999871 999870 999862 999862 999861 999860 999859 999857 999852 999846 999842 999839 999833 999832 999831 999830 999827 >
	999827 999827 999827 999825 999824 999824 999823 999820 999811 999805 999804 999803 999799 999798 999796 999795 999795 999791 >
	999791 999791 999789 999786 999781 999780 999779 999779 999771 999770 999766 999766 999763 999761 999761 999759 999756 999754 >
	999752 999751 999747 999745 999743 999741 999740 999739 999737 999736 999734 999731 999731 999730 999725 999724 999724 999722 >
	999720 999720 999720 999720 999717 999710 999708 999705 999704 999701 999700 999699 999699 999690 999689 999689 999686 999684 >
	999683 999682 999680 999678 999677 999674 999673 999672 999671 999662 999662 999662 999653 999653 999652 999651 999650 >
	999646 999644 999643 999641 999636 999634 999634 999629 999624 999620 999619 999614 999614 999611 999610 999610 999600 999599 >
	999596 999593 999593 999590 999588 999584 999579 999578 999578 999578 999577 999576 999571 999570 999566 999563 999558 999558 >
	999555 999552 999550 999549 999548 999547 999545 999545 999536 999536 999535 999532 999531 999530 999529 999526 999523 999522 >
	999521 999513 999510 999509 999506 999503 999499 999498 999496 999496 999496 999490 999490 999490 999482 999476 999473 999471 >
	999471 999470 999469 999469 999469 999467 999463 999461 999458 999455 999454 999449 999446 999444 999443 999443 999441 999438 >
	999436 999434 999431 999429 999428 999428 999428 999424 999422 999422 999420 999419 999416 999415 999409 999409 999408 999408 >

Результат работы кода на минимальных значениях:

input.txt:

1	1
2	1

output.txt:

1	h

	Время выполнения	Затраты памяти
Нижняя граница	0.002	25

диапазона значений входных данных из текста задачи		
Верхняя граница диапазона значений входных данных из текста задачи	0.019	30

Вывод по задаче:

В данной задаче мы изучили и реализовали алгоритм сортировки кучей.

Вывод

В данной лабораторной работе мы изучили структуру деревьев и сортировку кучей.

