Пирамидальная сортировка (HeapSort)

Бинарное дерево является пирамидой (Heap), если у каждого узла дерева нет сыновей с ключами большими (меньшими), чем ключ в этом узле.

Инвариант пирамиды H: $\forall b \in H$

$a[i]$ отец
$a[2i]$ левый сын $a[2i+1]$ правый сын

Анализ алгоритма:
Сложность в худшем случае $O(n \log_2 n)$

Поразрядная(распределяющая) сортировка

Пример с картами

Пример с числами:

Рассмотрим целые (положительные) трехразрядные числа в позиционной шестиричной системе счисления, то есть каждый разряд содержит цифры 0, 1, 2, 3, 4, 5

Например, пусть дана последовательность из 9 чисел: 203, 045, 141, 321, 522, 130, 054, 513

1 шаг — распределяем исла последовательности сначала по млдашей цифре по ящикам с «именами»: «0», «1», «2», «3», «4», «5».

Ящики заполняются снизу вверх.

1 шаг:
0 — 130
1 — 141, 321
2 — 522
3 — 203, 513
4 — 054
5 — 405, 045

Сцепляем последовательно слева направо содержимое ящиков в одну последовательность, так, что нижнее число правого ящика следует за верхним числом левого соседнего ящика (внутри ящиков последовательность расположена снизу вверх).

2 шаг:
130, 141, 321, 522, 203, 513, 054, 045, 405
Аналогичным образом распределяем по ящикам числа полученной последовательности по средней цифре.

0 — 405, 203
1 — 513
2 — 522, 321
3 — 130
4 — 045, 141

5 — 054
Повторяя сцепление содержимого ящиков, получим последовательность: 203, 405, 513, 321, 522, 130, 141, 045, 054, упорядоченную по двум младшим цифрам.

3 шаг:
203, 405, 513, 321, 522, 130, 141, 045, 054

Аналогичным образом распределяем по ящикам числа полученной последовательности по старшей цифре.

0 — 054, 045
1 — 141, 130
2 — 203
3 — 321
4 — 405
5 — 522, 513

Повторяя сцепление содержимого ящиков, получим упорядоченную последовательность: 045, 054, 130, 141, 203, 321, 405, 513, 522

Список — это абстрактный тип данных, представляющий собой упорядоченный набор значений, в котором некоторые значение может встречаться более одного раза.
Связный список — структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки («связки» на следующий и/или предыдущий узел списка)

Двусвязный список — по двусвязному списку можно передвигаться в любом направлении.
Кольцевой список — последний элемент содержит ссылку на первый. В этом списке константы NULL не существует.

Свойства связных списков:

- Размер списка — количество элементов в нём, исключая последний «нулевой» эемент, являющийся по определению пустым списком.
- Тип элементов — тот самый тип, над которым строится список, все элементы в списке должны быть этого типа.
- Отсортированность — список может быть отсортирован в соответствии с некоторыми критериями сортировки (например, по возрастанию целочисленных значений, если список состоит из целых чисел)
- Возможность доступа — некоторые списки в зависимости от реализации могут обеспечивать программиста селекторами для доступа непосредственно к заданному по номеру элементу.
- Сравниваемость — списки можно сравнивать друг с другом на соответствие, причем в зависимости от реализации операция сравнения списков может использовать разные технологии

Недостатки:

- Сложность определения адреса элемента по его индексу (номеру)в списке.
- На поля-указатели расходуется дополнительная память.
- Работа со списком медленнее, чем с массивами.
- Элементы списка могут быть расположены в памяти разреженно, что окажет. негативный эффект на кэширование процессора.

Достоинства:

- Лёгкость добавления и удаления элементов
- Размер ограничен только объёмом памяти и разрядностью указателей
- Динамическое добавление и удаление элементов

Стек (англ. stack — стопка) — структура данных, в которой доступ к элементам организован по принципу LIFO (англ. last in — first out, «последним пришёл первым вышел»).

Добавление элемента, называемое также проталкиванием (*push*), возможно только в вершину стека (добавленный элемент становится первым сверху). Удаление элемента, называемое также выталкиванием (*pop*), тоже возможно только из вершины стека, при этом второй сверху элемент становится верхним.

Очередь — структура данных с дисциплиной доступа к элементам первый пришёл - первый вышел.

Добавление в очередь возможно лишь в конец очереди, выборка - только из начала очереди., при этом выбранный элемент из очереди удаляется.

Ассоциативный массив - абстрактный тип данных, позволяющий хранить пары вида ключ, значение и добавления пары, а также поиска и удаления пары по ключу.

```
#include <stack>

int main() {
	stack st;
	into s = st.size();
	st.push(5);
	st.push(7);
	if (st.is empty()) {
		int c = st. pop();
	}
}
```