**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**Кафедра інформаційних систем**

**Алгоритми та складність**

**Завдання № 7**

**Звіт**

Артюхов Яків

Група К-28

**⦁ Умова завдання.**

Реалізувати біноміальну піраміду.

Біноміальна купа (англ. binomial heap) — це множина біноміальних дерев, що задовольняє властивостям біноміальної купи:

Кожне біноміальне дерево у купі підпорядковується властивості неспадної купи (англ. min-heap property): ключ вузла не менший за ключ його батьківського вузла.

Для будь якого невід'ємного цілого k в купі існує не більше одного біноміального дерева,чий корінь має ступінь k.

З даних властивостей випливає, що біноміальна купа, що має n вузлів, складається з не більше ніж [log(n)] + 1 біноміальних дерев.

Завдяки своїй структурі, біноміальне дерево ступеня k можна побудувати з двох дерев ступеня k−1 тривіальним приєднанням одного з них до іншого як найлівішого підпорядкованого дерева. Ця властивість є центральною для операції злиття біноміальних дерев, яка становить їхню основну перевагу над звичайними купами.

Ім'я походить від того факту, що біноміальне дерево ступеня n має (n, d) вузлів на глибині d.

**⦁ Опис алгоритму.**

*Операціі для роботи з біноміальною пірамідою:*

• MAKE\_HEAP(): створення нової порожньої піраміди.

• INSERT(H,x): вставка готового вузла x в піраміду H.

• MINIMUM(H): повертає вказівник на вузол піраміди H з найменшим ключем.

• EXTRACT\_MIN(H): видаляє вузол піраміди H з найменшим ключем і повертає вказівник на нього.

• UNION(H1,H2): повертає нову піраміду – результат злиття H1, H2 (вони не зберігаються).

• DECREASE\_KEY(H,x,k): присвоєння вузлу x піраміди H значення ключа k, що є меншим за поточне.

• DELETE(H,x): видалення вузла x з піраміди H.

*Представлення біноміальних пірамід*

• Кожне біноміальне дерево зберігається у представленні з лівим дочірнім та правим сестринським вузлами.

• Ключ key[x], вказівник на батька p[x], вказівник на найлівішого сина child[x], вказівник на правого брата sibling[x], кількість дочірніх вузлів degree[x].

• Біноміальна піраміда представлена списком коренів її біноміальних дерев впорядкованим за зростанням степенів дерев.

• Вказівник на перший корінь біноміальної піраміди H: head[H].

• Якщо x – корінь, то sibling[x] вказує на наступний корінь у списку.

**⦁ Аналіз алгоритму.**

*Операция Время работы*

*makeHeap O(1)*

*insert O(logn)*

*getMinimum O(logn)*

*extractMin Θ(logn)*

*merge Ω(logn)*

*decreaseKey Θ(logn)*

*delete Θ(logn)*

*insert*

*Чтобы добавить новый элемент в биномиальную кучу нужно создать биномиальную кучу H′ с единственным узлом, содержащим этот элемент, за время O(1) и объединить ее с биномиальной кучей H за O(logn), так как в данном случае куча H′ содержит лишь одно дерево.*

*extractMin*

*Приведенная ниже процедура извлекает узел с минимальным ключом из биномиальной кучи и возвращает указатель на извлеченный узел.*

*Процедура выполняется за время Θ(logn), поскольку всего в списке Θ(logn) корней биномиальных деревьев. И всего у найденного дерева k порядка (с минимальным значением ключа) ровно k детей, то сложность перебора этих детей будет тоже Θ(logn). А процесс слияния выполняется за Ω(logn). Таким образом, операция выполняется Θ(logn).*

*merge*

*Эта операция, соединяющая две биномиальные кучи в одну, используется в качестве подпрограммы большинством остальных операций.*

*Вот в чем состоит ее суть: пусть есть две биномиальные кучи с H и H′. Размеры деревьев в кучах соответствуют двоичным числам m и n, то есть при наличии дерева соответствующего порядка в этом разряде числа стоит единица, иначе ноль. При сложении столбиком в двоичной системе происходят переносы, которые соответствуют слияниям двух биномиальных деревьев Bk−1 в дерево Bk. Надо только посмотреть, в каком из сливаемых деревьев корень меньше, и считать его верхним (пример работы для одного случая приведен на рисунке справа; в другом случае подвешиваем наоборот).*

*decreaseKey*

*Следующая процедура уменьшает ключ элемента x биномиальной кучи, присваивая ему новое значение. Вершина, ключ которой был уменьшен, «всплывает» как в обычной куче. Процедура выполняется за время Θ(logn), поскольку глубина вершины x в худшем случае есть Θ(logn) (свойства биномиального дерева), а при выполнении каждого шага алгоритма мы поднимаемся вверх.*

*delete*

*Удаление ключа сводится к операциям decreaseKey и extractMin: сначала нужно уменьшить ключ до минимально возможного значения, а затем извлечь вершину с минимальным ключом. В процессе выполнения процедуры этот узел всплывает вверх, откуда и удаляется. Процедура выполняется за время Θ(logn), поскольку каждая из операций, которые используется в реализации, работают за Θ(logn).*