Алгоритми та структури даних
Модуль 2: Методи розробки алгоритмів
Лекція 11. Жадібні алгоритми.

Кудін Олексій Володимирович
avk256@gmail.com

План

1. Визначення
2. Алгоритм Прима.
3. Алгоритм Крускала.
4. Алгоритм Дейкстри.
5. Дерева Хаффмана.

Питання по попередній лекції

- *Яка основна ідея метода перетворення?*
- *Дайте визначення збалансованого дерева пошуку?*
- *Що таке 2-3 дерево пошуку?*
- *В чому відмінність піраміди від дерева пошуку?*

Етапи метода декомпозиції

- Перетворення в більш простий або більш зручний для рішення екземпляр тої ж задачі - спрощення екземпляра.
- Зміна подання наявного екземпляра задачі.
- Приведення задачі, тобто перетворення до екземпляра іншої задачі, для якої є алгоритм рішення.

Жадібний алгоритм

Жадібний алгоритм — це алгоритм, який приймає найкраще рішення, виходячи з наявних на поточному етапі даних, не турбуючись про можливі наслідки, сподіваючись врешті-решт отримати оптимальне рішення.

Жадібний алгоритм

Зазвичай, жадібний алгоритм базується на п'яти принципах:

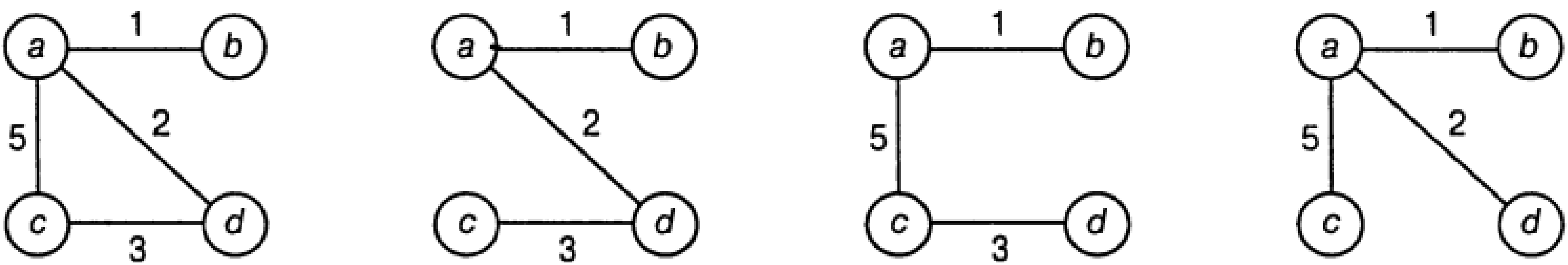
1. Набір можливих варіантів, з яких робиться вибір;
2. Функція вибору, за допомогою якої знаходиться найкращий варіант;
3. Функція придатності, яка визначає придатність отриманого набору;
4. Функція цілі, оцінює цінність рішення, не виражена явно;
5. Функція розв'язку, яка вказує на те, що знайдене кінцеве рішення.

Приклад 1. Задача розміну деякої суми монетами з відомим номіналом.

Кістякове дерево

Кістякове (каркасне) дерево – ациклічний за’язний підграф даного графа, який містить всі його вершини.

Мінімальне кістякове дерево у зв'язаному, зваженому, неорієнтованому графі — це кістяк цього графа, що має мінімальну можливу вагу, де під вагою дерева розуміється сума ваг його ребер.



Граф $w(T_1) = 6$ $w(T_2) = 9$ $w(T_3) = 8$

Алгоритм Прима

Алгоритм Прима - алгоритм побудови мінімального кістякового дерева зваженого зв'язного неорієнтованого графа.

Побудова починається з дерева, що включає в себе одну (довільну) вершину. Протягом роботи алгоритму дерево розростається, поки не охопить всі вершини вихідного графа. На кожному кроці алгоритму до поточного дереву приєднується найлегше з ребер, що з'єднують вершину з побудованого дерева і вершину, що не належить дереву.

Алгоритм $Prim(G)$

// Алгоритм Прима построения минимального остовного дерева
// **Входные данные:** Взвешенный связный граф $G = \langle V, E \rangle$
// **Выходные данные:** E_T, множество ребер, составляющих
// минимальное остовное дерево G

$V_T \leftarrow \{v_0\}$ // Множество вершин дерева инициализируется
// произвольной вершиной

$E_T \leftarrow \varnothing$

for $i \leftarrow 1$ **to** $|V| - 1$ **do**

Поиск ребра с минимальным весом $e^* = (v^*, u^*)$ среди всех ребер (v, u) таких, что $v \in V_T$ и $u \in V - V_T$

$V_T \leftarrow V_T \cup \{u^*\}$

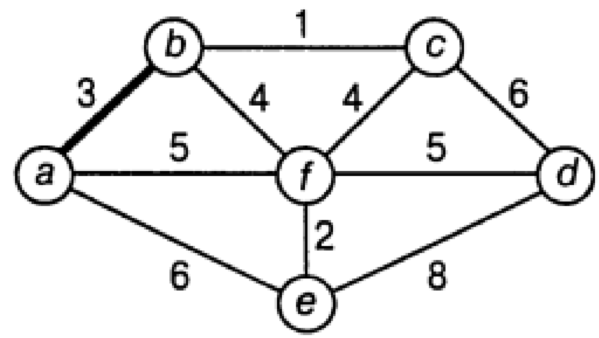
$E_T \leftarrow E_T \cup \{e^*\}$

return E_T

Обчислювальна складність – $O(|E|\log|V|)$

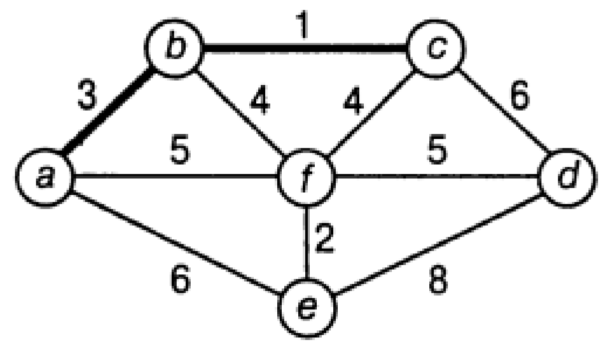
a(–,–)

b(a,3) c(–,∞) d(–,∞)
e(a,6) f(a,5)



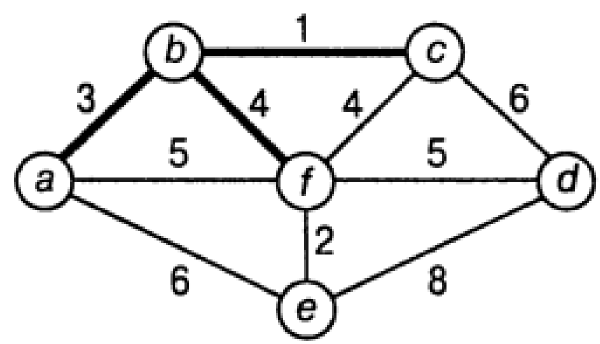
b(a,3)

c(b,1) d(–,∞) e(a,6)
f(b,4)



c(b,1)

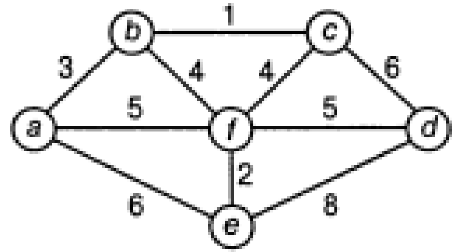
d(c,6) e(a,6) **f(b,4)**

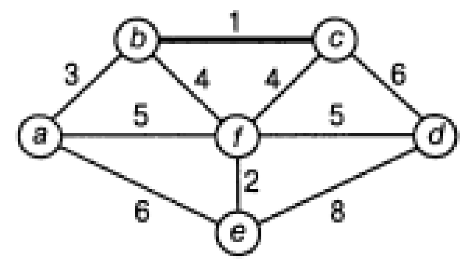
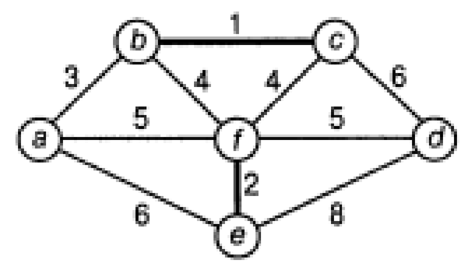
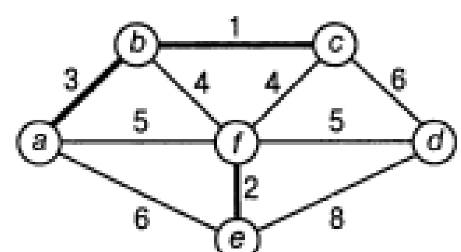


Алгоритм Крускала

Алгоритм Крускала починається з побудови виродженого лісу, що містить V дерев, кожне з яких складається з однієї вершини. Далі виконуються операції об’єднання двох дерев, для чого використовуються найкоротші можливі ребра, поки не утвориться єдине дерево. Це дерево і буде мінімальним кістяковим деревом.

Алгоритм Крускала

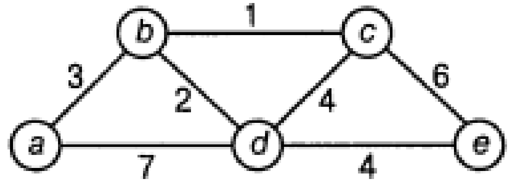


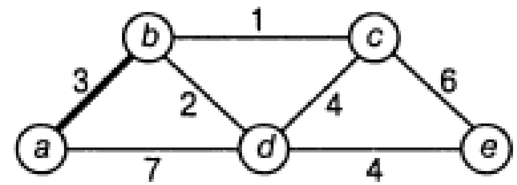
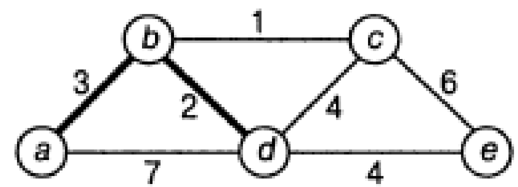
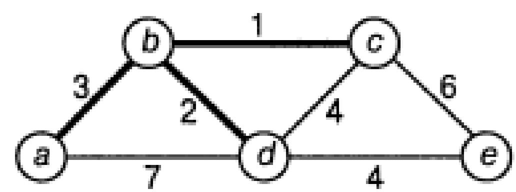
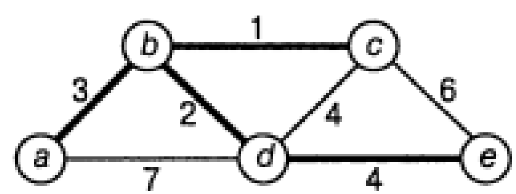
Вершины дерева	Отсортированный список ребер*	Рисунок
	bc ef ad bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
bc 1	bc **ef** ad bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
ef 2	bc ef **ad** bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	

Алгоритм Дейкстри

Алгоритм Дейкстри — алгоритм на графах, що знаходить найкоротший шлях від однієї вершини графа до всіх інших вершин. Класичний алгоритм Дейкстри працює тільки для графів без циклів від'ємної довжини.

Алгоритм Дейкстри



Вершины дерева	Остальные вершины	Рисунок
a(–,0)	**b(a,3)** c(–,∞) d(a,7) e(–,∞)	
b(a,3)	c(b,3+4) **d(b,3+2)** e(–,∞)	
d(b,5)	**c(b,7)** e(d,5+4)	
c(b,7)	**e(d,9)**	
e(d,9)		

Алгоритм Хаффмана

Алгоритм Хаффмана - адаптивний жадібний алгоритм оптимального префіксного кодування алфавіту з мінімальною надмірністю.

Цей метод кодування складається з двох основних етапів:

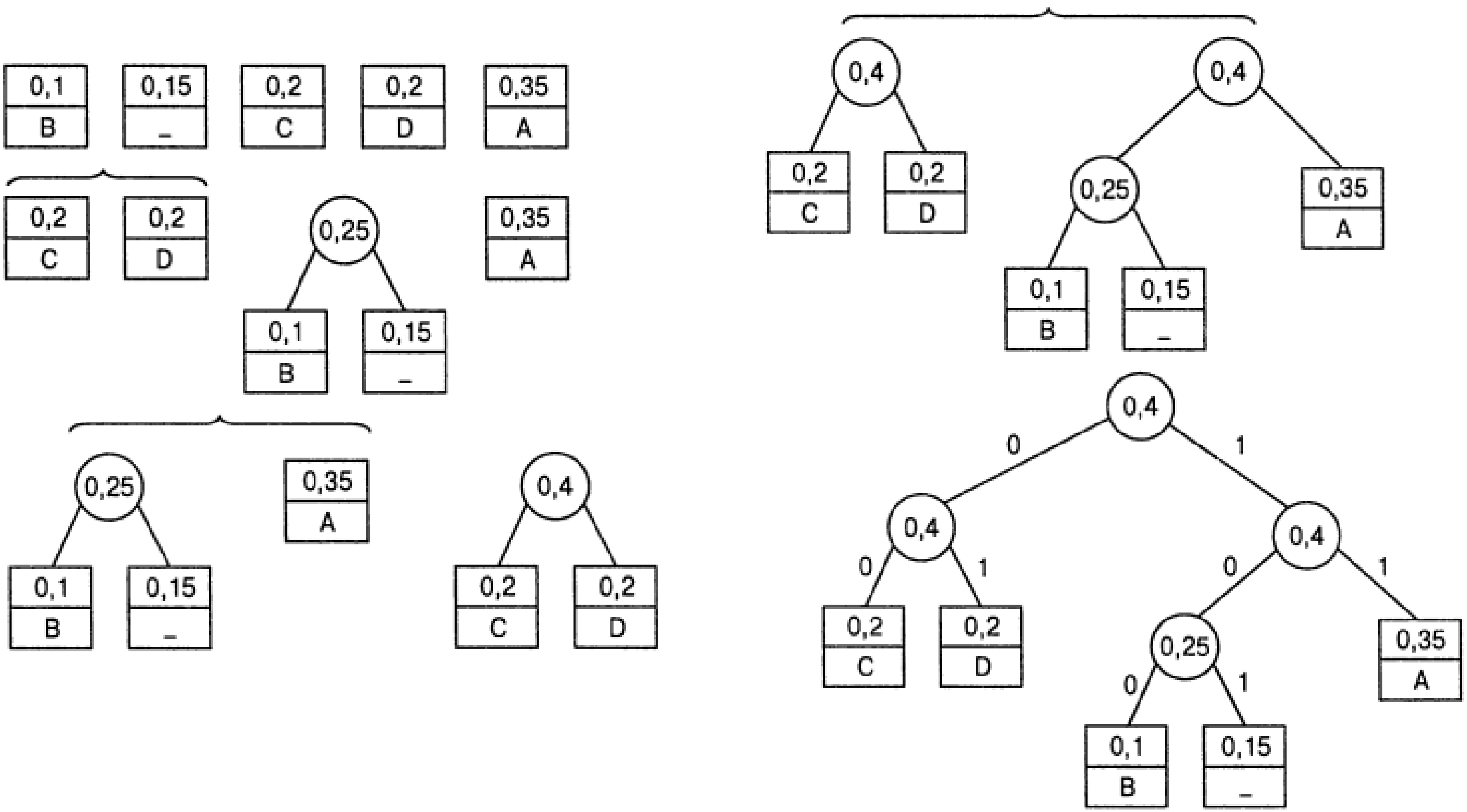
1. Побудова оптимального кодового дерева (H-дерево).
2. Побудова відображення коду-символів на основі побудованого дерева.

Алгоритм Хаффмана. Побудова Н-дерева

1. Символи вхідного алфавіту утворюють список вільних вузлів. Кожен лист має вагу, яка може бути рівною або ймовірності, або кількості входжень символу у стиснене повідомлення.
2. Вибираються два вільних вузла дерева з найменшими вагами.
3. Створюється їх батьковий вузол з вагою, рівною їх сумарній вазі.
4. Вузол-батько додається в список вільних вузлів, а два його нащадка видаляються з цього списку.
5. Одній дузі, котра виходить з вузла батька, ставиться у відповідність біт 1, інший — біт 0.
6. Кроки, починаючи з другого, повторюються доти, поки в списку вільних вузлів не залишиться тільки один вільний вузол. Він і буде вважатися коренем дерева.

Алгоритм Хаффмана. Побудова Н-дерева

Символ	*A*	*B*	*C*	*D*	_
Вероятность	0.35	0.1	0.2	0.2	0.15
Код	11	100	00	01	101



Резюме (що встигли за лекцію)

- Дали загальне визначення жадібного методу розробки алгоритмів.
- Розглянули Прима та Крускала побудови мінімальних каркасних дерев.
- Розглянули алгоритм Дейкстри побудови найкоротшого шляху.
- Розглянули алгоритм Хаффмана стиснення даних.

До наступної лекції:

Повторити всі розглянуті алгоритмічні задачі та методи розробки алгоритмів для їх розв'язання.