Лекція 7.
Алгоритми кодування інформації за допомогою дерев.

§1. Стиснення (Кодування) даних

Стиснення даних – це процес, який забезпечує зменшення об'єму даних за рахунок зміни способу їх організації.

Всі алгоритми стиснення оперують вхідним потоком інформації, мінімальною одиницею якої є біт, а максимальної - кілька біт, байт або кілька байт.

Метою процесу стиснення, як правило, є отримання більш компактного вихідного потоку інформаційних одиниць з деякого спочатку некомпактного вхідного потоку за допомогою деякого їх перетворення.

Основними технічними характеристиками процесів стиснення і результатів їх роботи є:

- ступінь стиснення (compress rating) або відношення (ratio) обсягів вихідного і результуючого потоків;
- швидкість стиснення - час, що витрачається на стиснення деякого обсягу інформації вхідного потоку, до отримання з нього еквівалентного вихідного потоку;
- якість стиснення - величина, що показує на скільки сильно упакований вихідний потік, за допомогою застосування до нього повторного стиснення з цього ж або іншому алгоритму.

§2. Кодування Хаффмана

2.1 Ідея алгоритму Хаффмана

Один з перших алгоритмів ефективного кодування інформації був запропонований Д. А. Хаффманом в 1952 році. Ідея алгоритму полягає в наступному: знаючи ймовірності входження символів в повідомленні, можна описати процедуру побудови кодів змінної довжини.

Символам з більшою ймовірністю присвоюються більш короткі коди. Коди Хаффмана володіють властивістю префіксності, що дозволяє однозначно їх декодувати. Класичний алгоритм Хаффмана на вході отримує таблицю частот входженнь символів у повідомленні. Далі на підставі цієї таблиці будується дерево кодування Хаффмана (Н-дерево).

2.2 Коди фіксованої та змінної довжини

Нехай є файл даних, який треба стиснути. Файл містить 6 символів a, b, c, d, e, f, які зустрічаються з різною частотою.

Розглянемо задачу по розробці бінарного коду символів, в якому кожний символ представляється унікальним бінарним рядком. Якщо використовувати код **фіксованої довжини** (рівномірний код), то для представлення кожного з 6 символів знадобиться 3 біти:

Символ	a	b	c	d	e	f
Частота входження	45	13	12	16	9	5
Код фіксованої довжини	000	001	010	011	100	101

При використанні такого методу для кодування всього файлу знадобиться: 100 · 3біти=300 бітів.

За допомогою коду змінної довжини (нерівномірного коду) можна отримати значно кращі результати. Це досягається за рахунок того, що символу, який зустрічається найчастіше, співставляється коротке кодове слово, а символу, що зустрічається рідко – довге.

Символ	a	b	c	d	e	f
Частота входження	45	13	12	16	9	5
Код фіксованої довжини	000	001	010	011	100	101
Код Змінної довжини	0	101	100	111	1101	1100

При використанні такого методу для кодування всього файлу знадобиться:

$$45 \cdot 1+13 \cdot 3+12 \cdot 3+16 \cdot 3+9 \cdot 4+5 \cdot 4=224.$$

Завдяки цьому економиться 25% об'єму.

2.3 Префіксні коди

Будемо розглядати тільки ті коди, в яких ніяке кодове слово не є префіксом будь-якого іншого кодового слова. Такі коди називаються **префіксними**.

Наприклад, при кодуванні за допомогою префіксного коду змінної довжини трьохсимвольний файл abc має вигляд 0 101 100.

Такі коди спрощують декодування. Початкове кодове слово легко ідентифікувати, перетворивши його у вхідний символ, потім наступне кодове слово і т.д.

Наприклад, декодувати рядок 001011101.

a	b	c	d	e	f
0	101	100	111	1101	1100

Відповідь: aabe

2.4 Алгоритм кодування

1) Символи вхідного алфавіту утворюють список вільних вузлів. Кожен лист має вагу, яка може дорівнювати або ймовірності, або кількості входжень символу в повідомлення.

2) Вибираються два вільних вузла дерева з найменшими вагами.

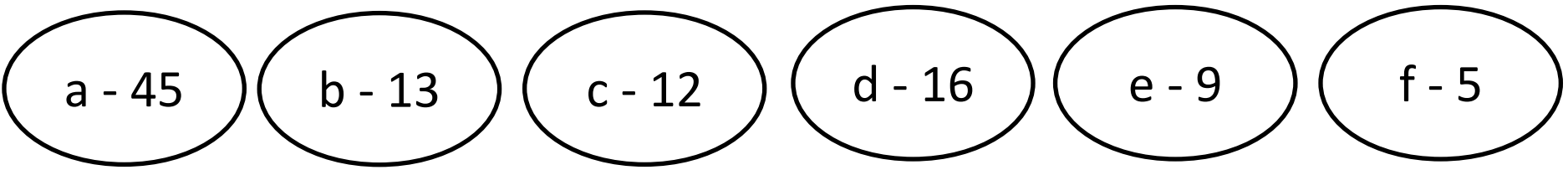
3) Створюється їх батько з вагою, рівною їх сумарній вазі. Батько додається в список вільних вузлів, а двоє його дітей (нащадків) видаляються з цього списку.

4) Правій гілці батьківського вузла, ставиться у відповідність біт 1, лівій - біт 0.

Кроки, починаючи з другого, повторюються доти, доки в списку вільних вузлів не залишиться жодного вузла. Корінь дерева дорівнюватиме загальній сумі всіх символів

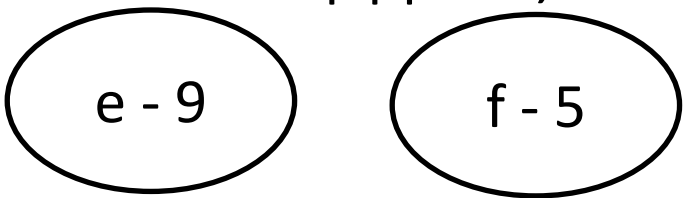
Приклад. Нехай є файл даних, який треба стиснути. Файл містить 6 символів a, b, c, d, e, f, які зустрічаються з різною частотою.

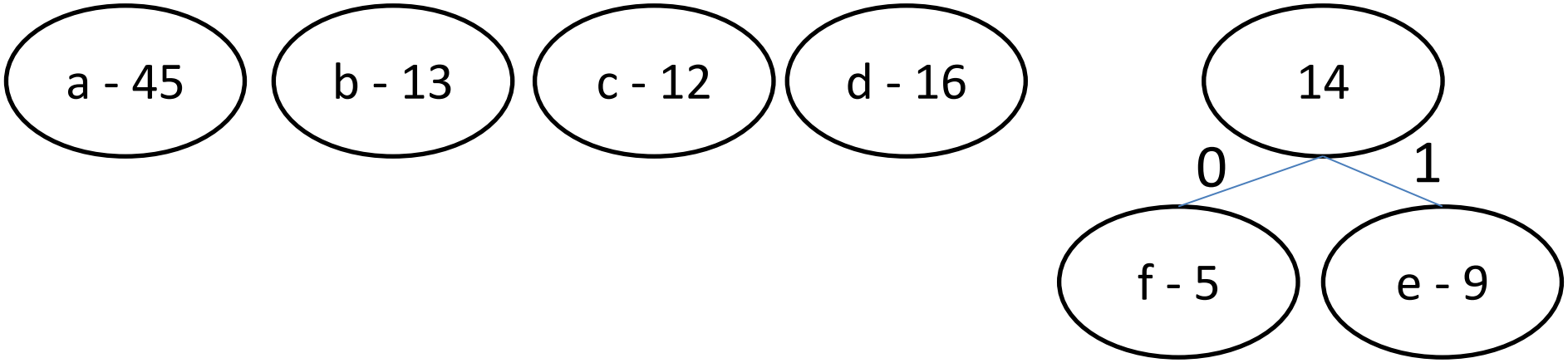
Символ	a	b	c	d	e	f
Частота входження	45	13	12	16	9	5



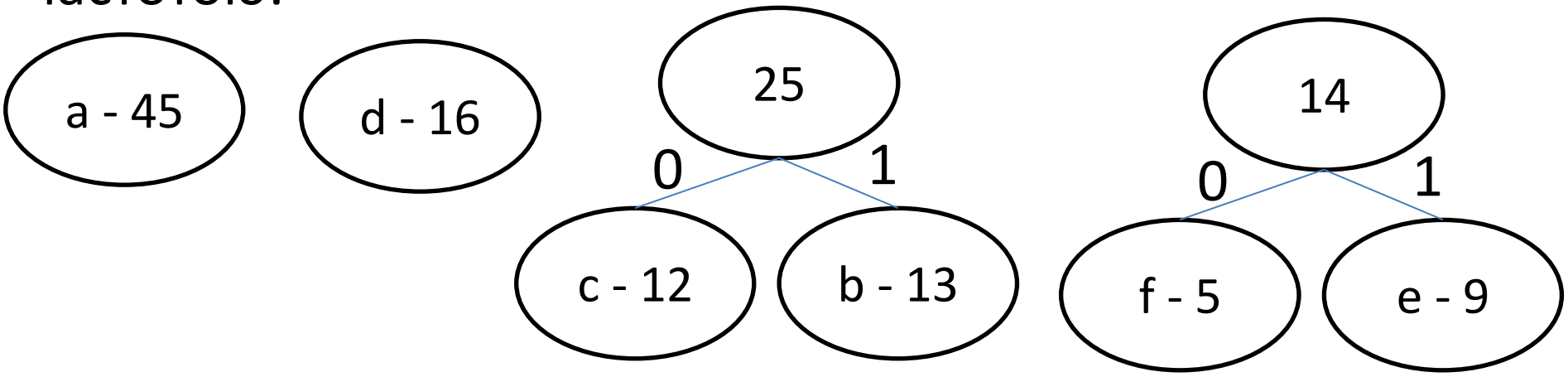
Обираємо два листа з найменшою частотою і комбінуємо їх в піддерево з вагою у батьківському вузлі, рівною сумі частот цих листів. Лист з більшою частотою буде правим нащадком, з меншою – лівим.

Це листи

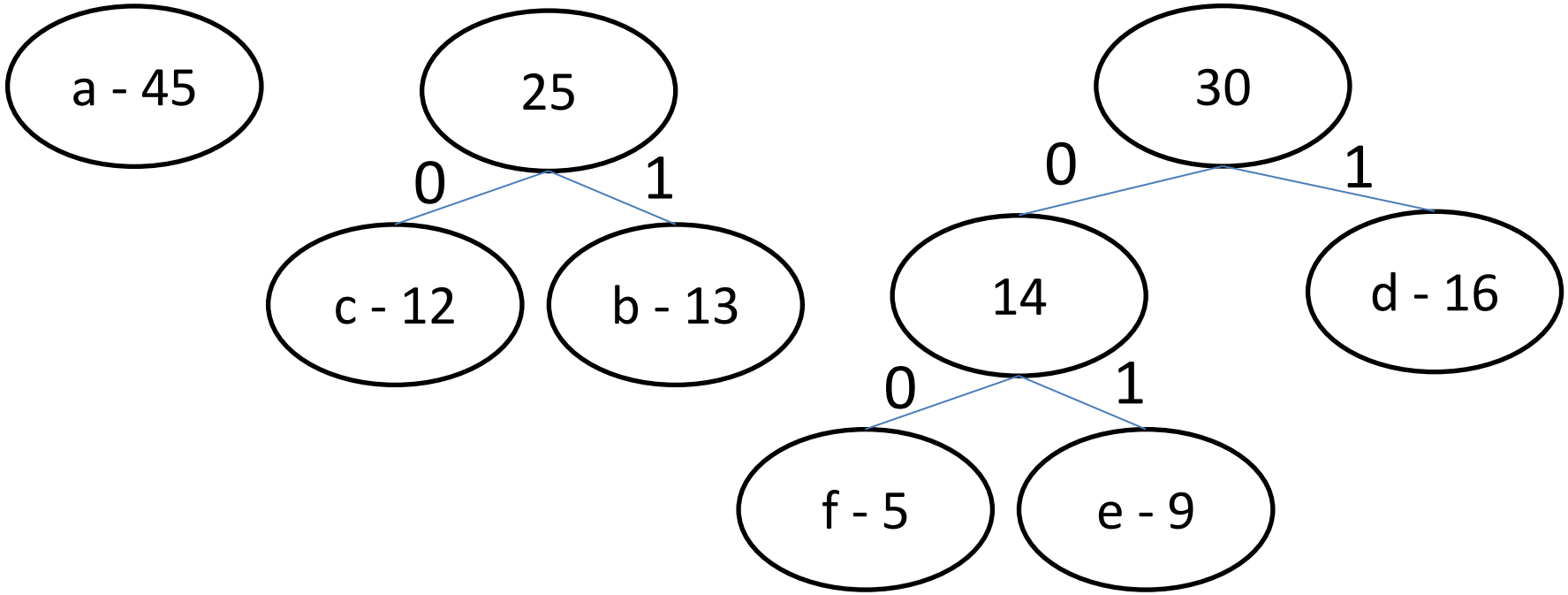




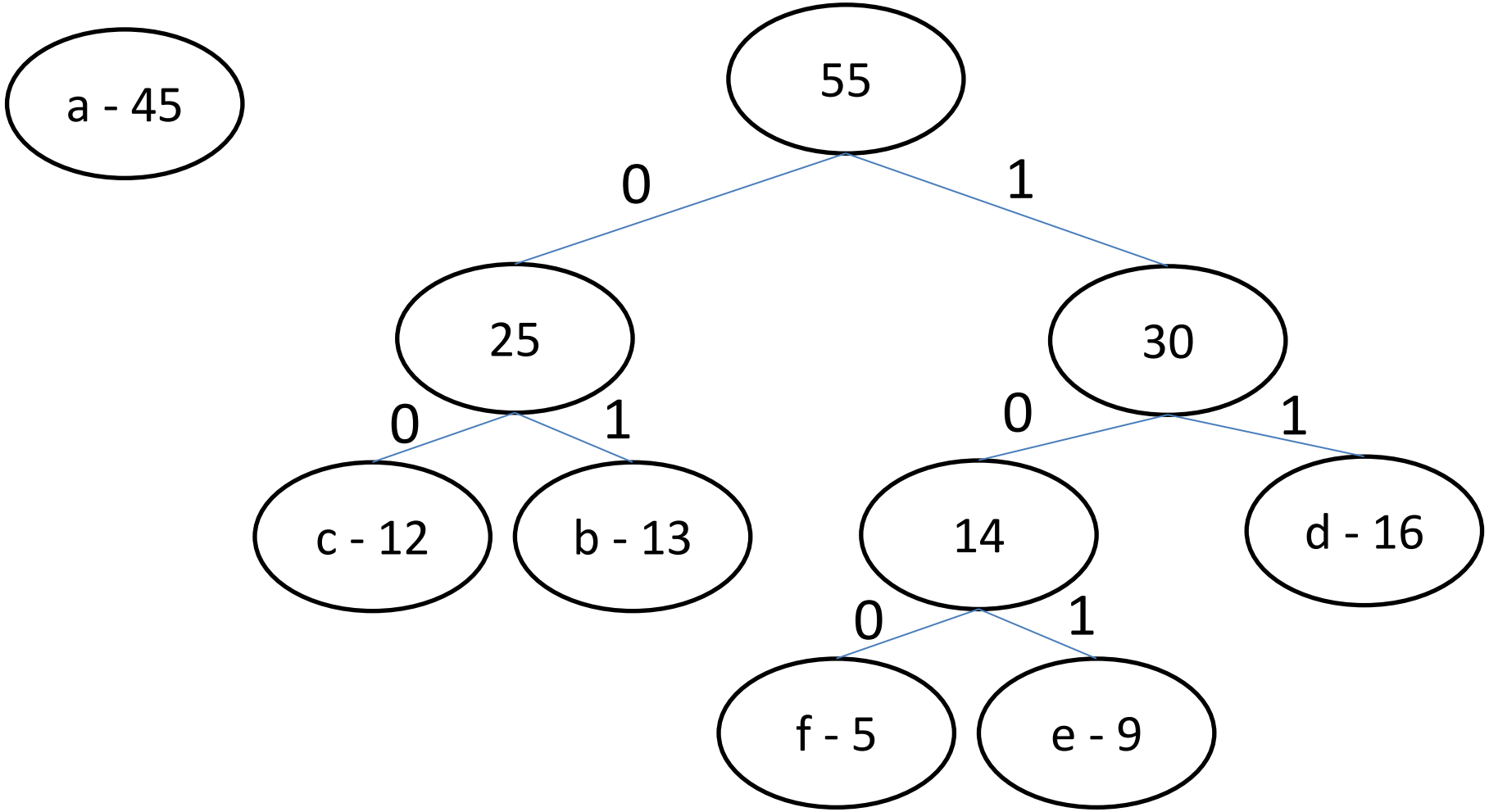
З утвореного лісу обираємо два елементи з найменшою частотою:



З утвореного лісу обираємо два елементи з найменшою частотою:

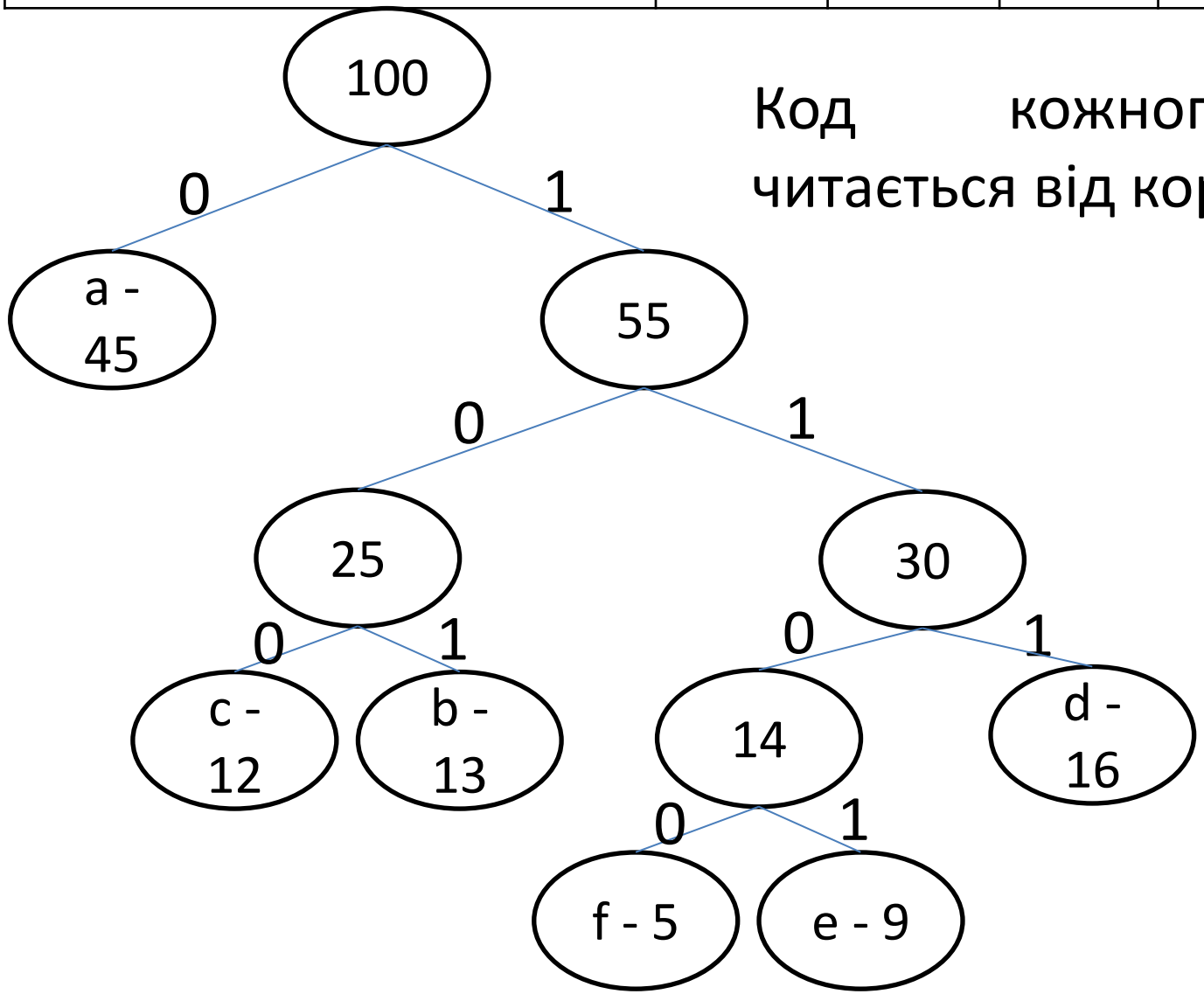


З утвореного лісу обираємо два елементи з найменшою частотою.



Поєднуємо два останніх елементи:

Символ	a	b	c	d	e	f
Код змінної довжини	0	101	100	111	1101	1100



Код кожного символа читається від кореня до листа.

§3. Алгоритм Шеннона-Фано

3.1 Ідея алгоритму Шеннона-Фано

Алгоритм Шеннона-Фано — один з перших алгоритмів стиснення, який сформулювали американські вчені Шеннон і Фано.

Даний метод стиснення має велику схожість з алгоритмом Хаффмана, який з'явився на кілька років пізніше.

Алгоритм використовує коди змінної довжини: символ, який часто зустрічається, кодується кодом меншої довжини, а той що рідше зустрічається — кодом більшої довжини. Коди Шеннона-Фано префіксні, тобто жодне кодове слово не є префіксом будь-якого іншого. Ця властивість дозволяє однозначно декодувати будь-яку послідовність кодових слів.

3.2 Алгоритм кодування

1) Символи первинного алфавіту сортують в порядку зменшення ймовірностей (частот).

2) Символи отриманого алфавіту ділять на дві частини, сумарні ймовірності (частоти) символів яких максимально близькі один одному.

3) У префіксному коді для першої частини алфавіту присвоюється цифра «0», другої частини — «1».

4) Отримані частини рекурсивно діляться і їх частинам призначаються відповідні двійкові цифри в префіксному коді.

Коли розмір підалфавіту стає рівним нулю або одиниці, то наступне подовження префіксного коду для відповідних йому символів первинного алфавіту не відбувається, таким чином, алгоритм привласнює різним символам префіксні коди різної довжини.

Код Шеннона-Фано будується за допомогою дерева. Побудова цього дерева починається від кореня. Вся множина кодованих елементів відповідає кореню дерева (вершині першого рівня). Воно розбивається на дві підмножини з приблизно однаковими сумарними ймовірностями. Ці підмножини відповідають двом вершинам другого рівня, які з'єднуються з коренем.

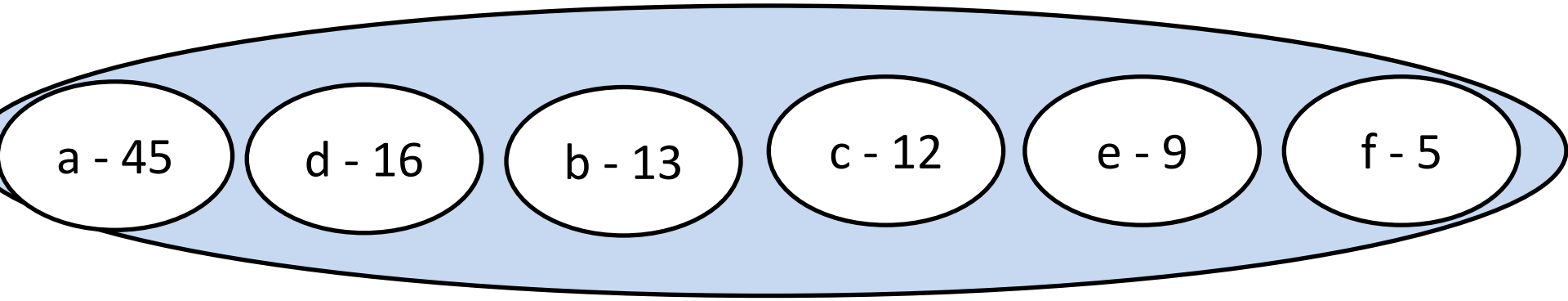
Далі кожна з цих підмножин розбивається на дві підмножини з приблизно однаковими сумарними ймовірностями. Їм відповідають вершини третього рівня. Якщо підмножина містить єдиний елемент, то йому відповідає кінцева вершина кодового дерева; така підмножина розбиттю не підлягає. Подібним чином поступаємо до тих пір, поки не отримаємо всі кінцеві вершини. Гілки кодового дерева розмічаємо символами 1 і 0, як у випадку коду Хаффмана.

При побудові коду Шеннона-Фано розбиття множини елементів може бути обрано, взагалі кажучи, декількома способами. Вибір розбиття на рівні *n* може погіршити варіанти розбиття на наступному рівні (*n* + 1) і призвести до неоптимальності коду в цілому. Іншими словами, оптимальна поведінка на кожному кроці шляху ще не гарантує оптимальності всієї сукупності дій. Тому код Шеннона-Фано не є оптимальним в загальному сенсі, хоча і дає оптимальні результати при деяких розподілах ймовірностей. Для одного і того ж розподілу ймовірностей можна побудувати, взагалі кажучи, кілька кодів Шеннона-Фано, і всі вони можуть дати різні результати. Якщо побудувати всі можливі коди Шеннона-Фано для даного розподілу ймовірностей, то серед них будуть знаходитися і всі коди Хаффмана, тобто оптимальні коди.

Приклад. Нехай є файл даних, який треба стиснути. Файл містить 6 символів a, b, c, d, e, f, які зустрічаються з різною частотою.

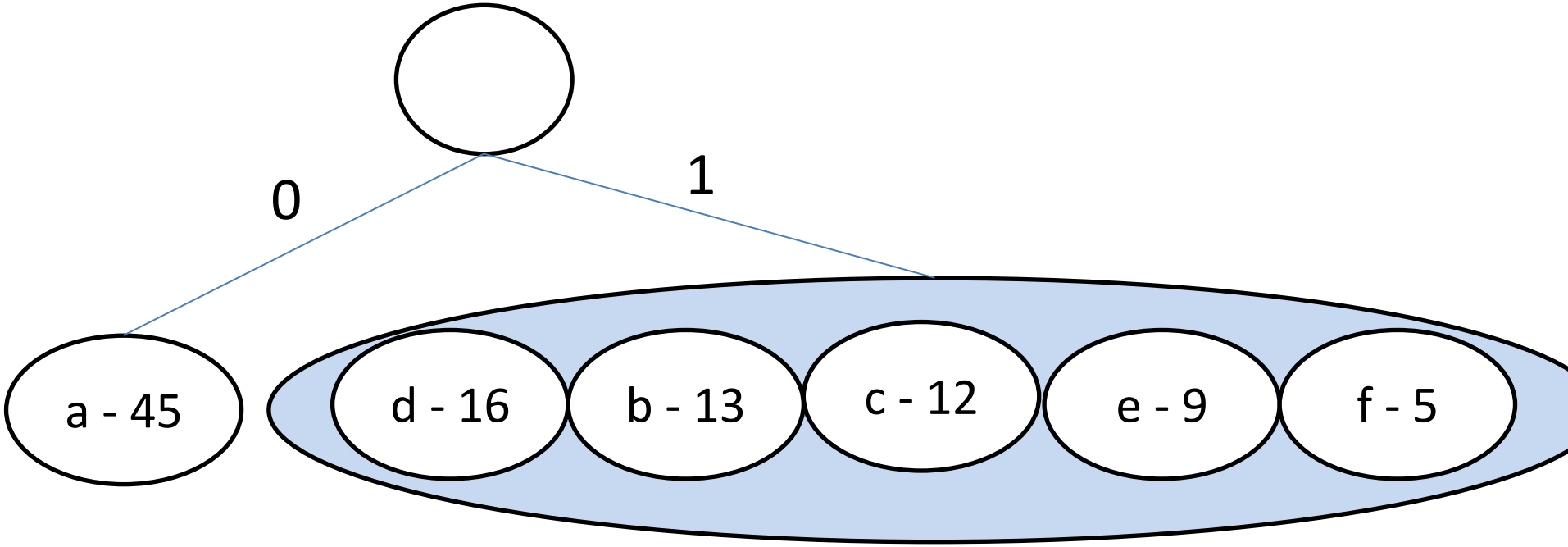
Символ	a	b	c	d	e	f
Частота входження	45	13	12	16	9	5

Вся множина кодованих елементів відповідає кореню дерева (вершині першого рівня).

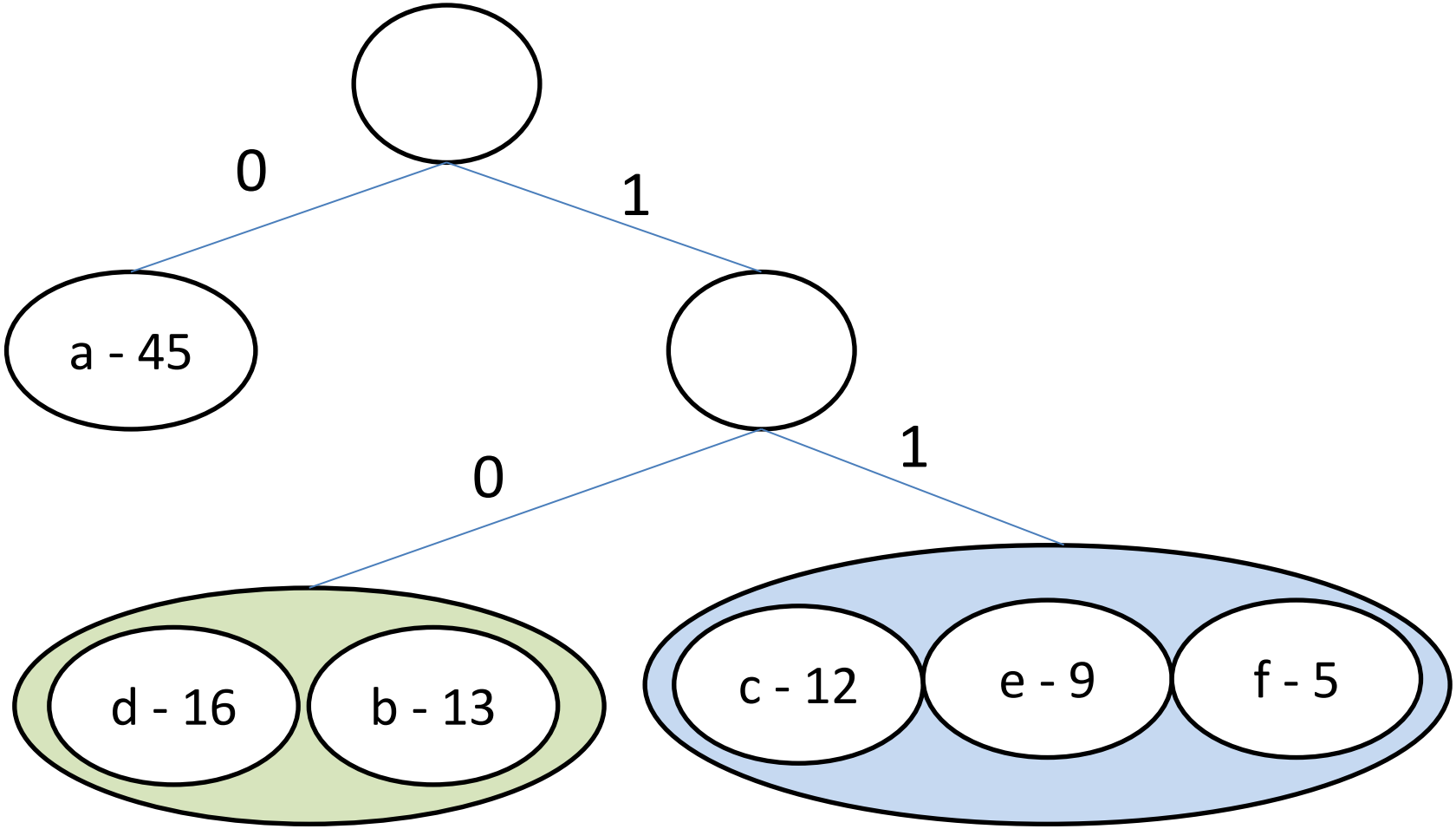


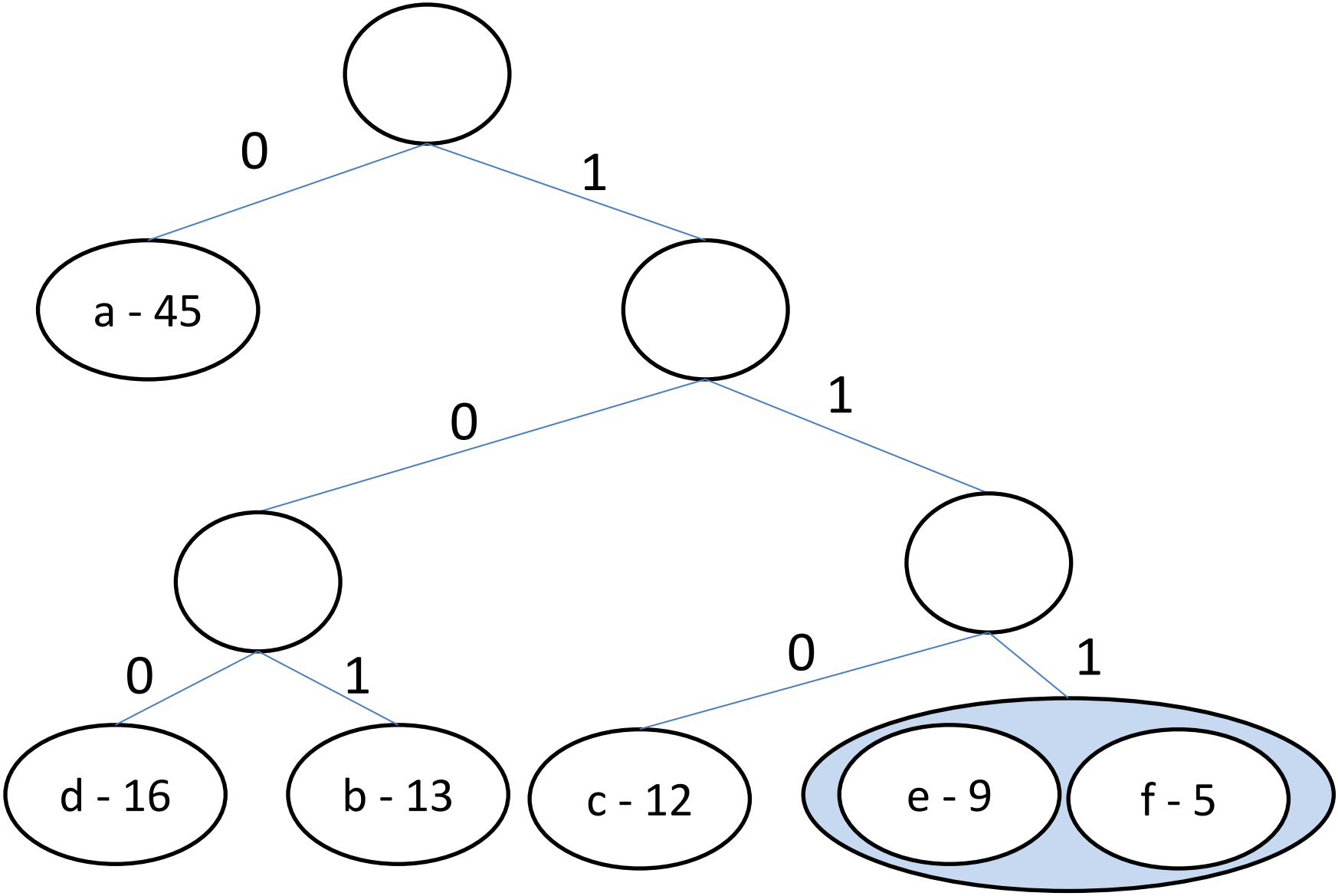
Розіб'ємо на дві підмножини з приблизно однаковими сумарними ймовірностями: 45 та 55.

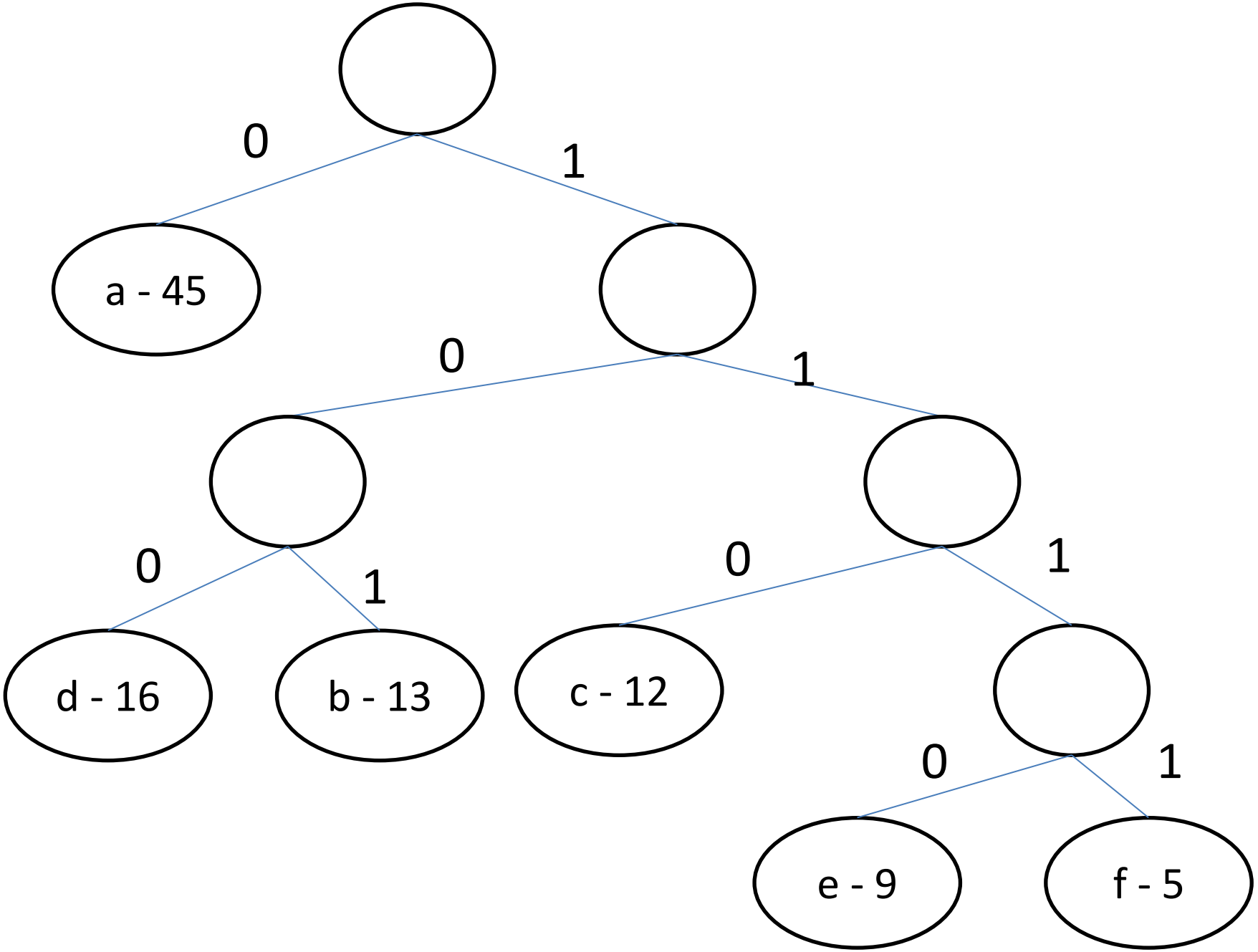
Ці підмножини відповідають двом вершинам другого рівня, які з'єднуються з коренем.



Розіб'ємо на дві підмножини з приблизно однаковими сумарними ймовірностями: 29 та 26.


0
1
a - 45
0
1
d - 16
b - 13
c - 12
e - 9
f - 5


0
1
a - 45
0
1
0
1
0
1
d - 16
b - 13
c - 12
e - 9
f - 5


0
1
a - 45
0
1
0
1
0
1
d - 16
b - 13
c - 12
0
1
e - 9
f - 5

Символ	a	b	c	d	e	f
Частота входження	45	13	12	16	9	5
Код	0	101	110	100	1110	1111

Також алгоритм Шеннона-Фано можна представити таблицею:

a – 45	0				0
d - 16	1	0	0		100
b - 13			1		101
c - 12		1	0		110
e - 9			1	0	1100
f - 5				1	1111