Kodowanie informacji za pomocą algorytmu Huffmana

W kodowaniu stosuje się różne techniki. W najprostszych zakłada się, że każdy znak koduje się za pomocą stałej liczby bitów. Z tej zasady korzysta kodowanie ASCII. W celu skrócenia zapisu (liczby bajtów w pliku) dopuszcza się do sytuacji, że różne znaki mogą być zakodowane za pomocą różnej liczby bitów. Warunkiem jest uzyskanie kodu **jednoznacznie dekodowalnego**, czyli takiego, w którym każdy ciąg kodowy ma tylko jedną odpowiadającą mu informację. Innymi słowy, nie da się odczytać innej wiadomości niż ta, która została zapisana w pliku. Powszechnie stosuje się w tym celu algorytm Huffmana (wersja zachłanna, wykorzystywana w kompresji JPEG lub MP3) albo algorytm Shannona-Fano, który wykorzystywany jest np. w kompresji ZIP.

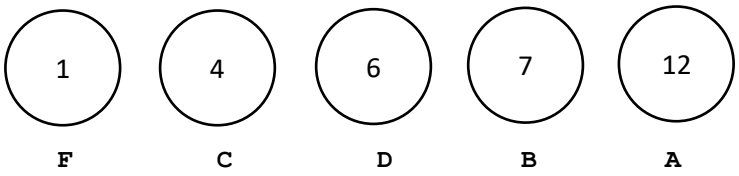
Powiedzmy, że chcemy zapisać informację zawartą w ciągu 30 znaków zawierających dla ułatwienia wyłącznie 5 wielkich liter alfabetu (w tym przypadku spacje są tylko po to, by ułatwić czytanie): **DBDB CADA ABDA DABA BAAC FABC ABAC DA**. W kodowaniu ASCII dla każdego znaku potrzebujemy 8 bitów, czyli łącznie byłoby potrzebne 240 bitów. Czy można zaoszczędzić bity dla zapisania w sposób jednoznacznie dekodowalny całej informacji?

Najpierw zauważmy, że dla rozróżnienia tego tekstu, składającego się z 5 liter, potrzebujemy tylko 3 bity, zatem do zakodowania całej informacji wystarczy już tylko 90 bitów. Czy można to zrobić jeszcze oszczędniej? Przeanalizujmy częstość występowania każdej litery:

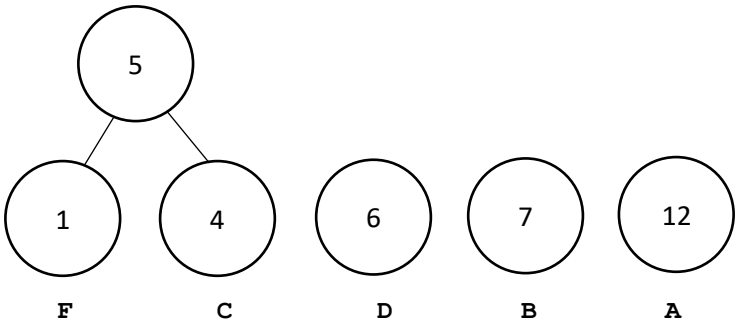
```
A – 12
B – 7
C – 4
D – 6
F – 1
```

Algorytm Huffmana polega na przyporządkowaniu symbolom najczęściej występującym najkrótszego słowa kodowego i odwrotnie, tj. symbolom najrzadziej występującym najdłuższego słowa kodowego. Najlepiej więc zakodować literę A na jednym bicie, bo wtedy uzyskamy najlepszy współczynnik kompresji. Pozostałe litery muszą być zakodowane na większej, lecz różnej liczbie bitów.

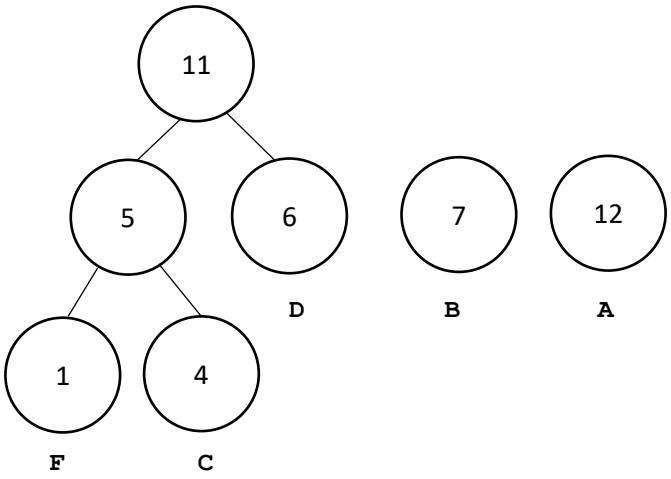
Algorytm Huffmana zakłada, że najpierw układamy wszystkie litery (lub inne symbole) w kolejności częstości ich wystąpień od najmniejszej do największej, przy czym symbol najrzadziej występujący znajduje się po lewej stronie, zaś najczęściej po prawej.

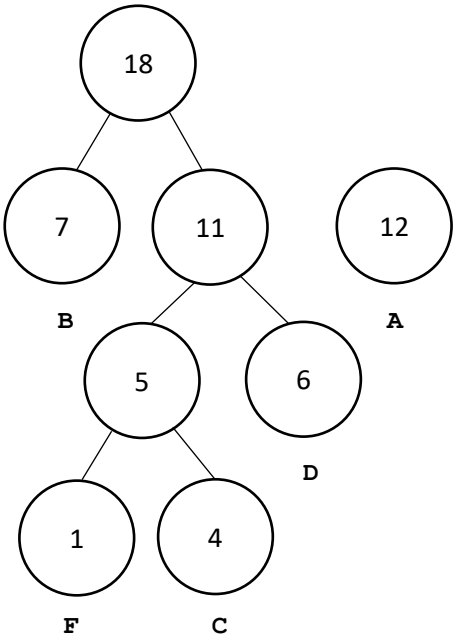


Następnie dwa skrajne lewe symbole (litery) zastępujemy jednym wspólnym węzłem, którego częstość występienia będzie sumą wystąpień tworzących go symboli. Na liście symboli pozostanie w ten sposób $n-1$ symboli, gdzie n to liczba symboli przed dokonaniem połączenia.

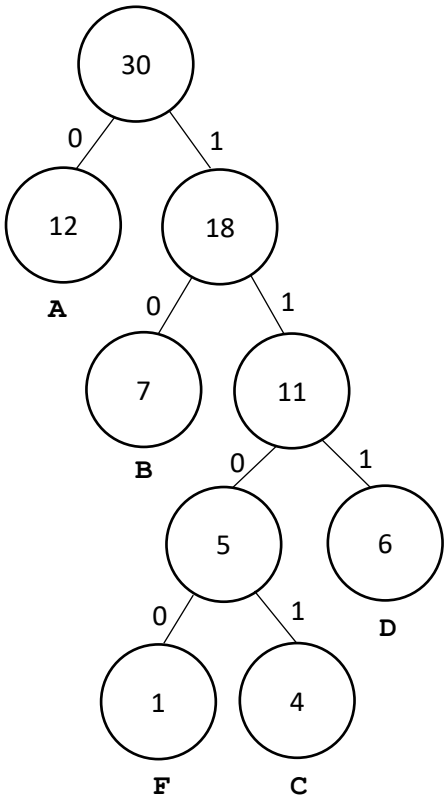


Nowy węzeł należy potraktować jako zwykły symbol. Powstałą w ten sposób nową listę należy ponownie posortować w kolejności niemalejącej i znowu skrajne symbole zastąpić nowym węzłem. Algorytm należy powtarzać do wyczerpania wszystkich symboli (aż pozostanie tylko jeden nowy węzeł, zwany korzeniem). Poniżej przedstawiono graficznie kolejne kroki algorytmu.





Przed nami ostatni krok algorytmu. Dodatkowo przyjmijmy teraz, że gałąź skierowana np. w lewo ma wartość bitową 0, a gałąź skierowana w prawo ma wartość bitową 1. To przyda nam się za chwilę, gdy będziemy ustalali wartość bitową dla każdej litery osobno, przechodząc do niej od korzenia. Całość będzie miała więc następujący wygląd.



Dzięki schematowi graficznemu możemy sporządzić tabelkę podsumowującą nasze działania:

Symbol	Kod dla danego symbolu	Długość kodu dla danego symbolu	Liczba wystąpień symbolu	Iloczyn długości kodu i liczby wystąpień symbolu
A	0	1	12	12
B	10	2	7	14
C	1101	4	4	16
D	111	3	6	18
F	1100	4	1	4
			SUMA	**64**

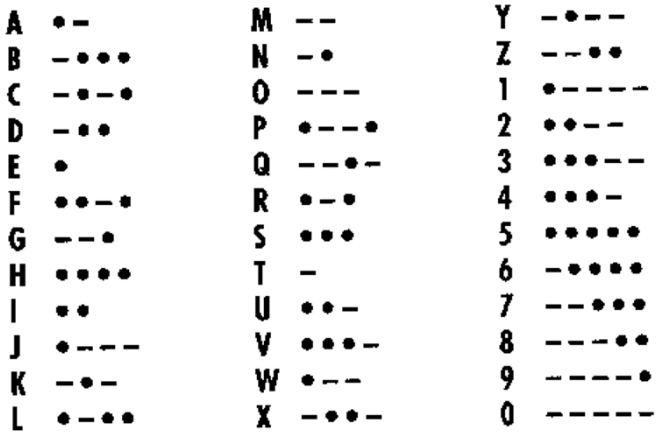
Zatem ostatecznie całą zakodowaną przez nas informację **DBDB CADA ABDA DABA BAAC FABC ABAC DA** można zapisać minimalnie za pomocą 64 bitów w następujący sposób:

1111011110 110101110 0101110 1110100 10001101 11000101101 01001101 1110

Ćwiczenie do wykonania na zajęciach – tylko grupa A

C1.

W alfabecie Morse'a litery i znaki przestankowe są zapisywane za pomocą ciągu kropek i kresek. W poniższej tabeli znajduje się fragment tego alfabetu. Opracuj program, który poprosi użytkownika o podanie ciągu tekstowego (bez polskich znaków diakrytycznych i bez spacji), a następnie zamieni go na ciąg tekstowy zapisany za pomocą alfabetu Morse'a, który przedstawiono poniżej.



Zadanie domowe – do wyboru Z1 albo Z1a – tylko dla chętnych

Z1.

Na podstawie powyższych informacji napisz w języku C# (Python lub C++) program, który zakoduje informację przedstawioną w zadaniu za pomocą opisanego algorytmu Huffmana.

Z1a.

Napisz program w wersji rozbudowanej, który pobiera z pliku TXT tekst zawierający małe i wielkie litery alfabetu angielskiego, cyfry, przecinki, kropki oraz spacje i koduje go do osobnego pliku tekstowego. Ponadto program powinien umożliwiać zdekodowanie ciągu bitów w pliku do oryginalnego tekstu.

Termin nadesłania rozwiązania: **31 maja 2023 r. (do północy)**