1. **Problem**

Dany jest zbiór oraz kolekcja jego podzbiorów. Należy podzielić zbiór na dwa rozłączne podzbiory i tak, aby żaden z podzbiorów nie zawierał się ani w , ani w .

1. **Koncepcja rozwiązania**

Przed rozpoczęciem próby znalezienia rozwiązania na podstawie otrzymanych danych istnieje możliwość odrzucenia najbardziej oczywistego przypadku, który nie ma rozwiązania – czyli takiego, gdzie jakikolwiek podzbiór ma tylko jeden element. Zakładając, że otrzymany zestaw danych nie został odrzucony podczas testu można przejść do rozwiązania.

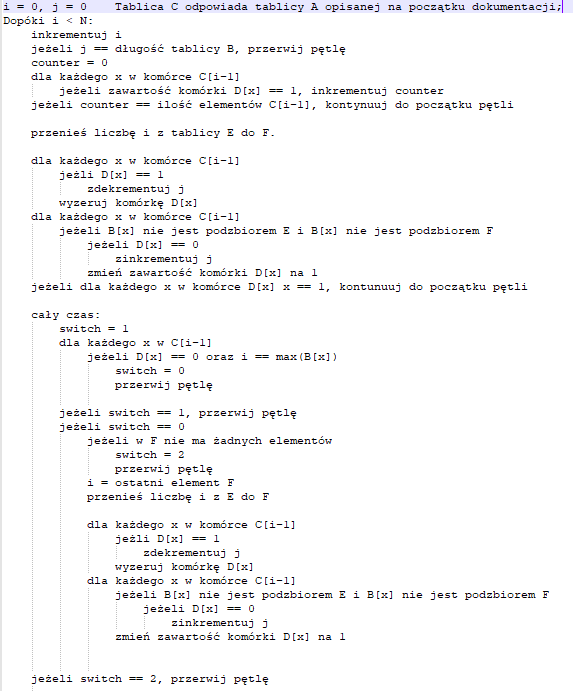
Najprostszym sposobem znalezienia rozwiązania problemu jest sprawdzenie wszystkich możliwych podziałów zbioru na dwa podzbiory. Jednakże ich ilość w liczbie ogranicza taki sposób do jedynie niedużych przypadków.

O wiele skuteczniejsza jest następująca metoda (*zakładamy w niej, że zbiór zawiera w sobie kolejne liczby od 1 do . Jeśli tak nie jest, to dla poprawnego działania tworzymy kopię zbioru , gdzie przeskalowujemy kolejne wartości na liczby . Po zakończeniu działania algorytmu, zamieniamy liczby z powrotem na wejściowe*):

1. Tworzymy listę o elementach, gdzie to ilość elementów zbioru , gdzie każdy indeks odpowiada poszczególnej liczbie ze zbioru S.
2. Tworzymy listę o elementach, gdzie to ilość elementów kolekcji podzbiorów zbioru .
3. Każdy kolejny element listy zawiera kolejny podzbiór od .
4. Każdy kolejny element listy zawiera w sobie listę liczb będących indeksami listy w taki sposób, że jeśli podzbiór zawiera w sobie liczbę , to element listy o indeksie będzie zawierać liczbę (wszystkie wartości pomniejszone o 1, bo indeksujemy od 0).
5. Tworzymy listę o elementach. Każdy element domyślnie ma wartość . Jeśli wynosi , oznacza to, że odpowiadający mu podzbiór spełnia założenia problemu.
6. Tworzymy pustą listę , która będzie zawierać w sobie wartości tych indeksów tablicy , dla których liczba , będzie umieszczona w zbiorze .
7. Tworzymy listę , która będzie zawierać w sobie wartości wszystkich indeksów tablicy , dla których liczba (, będzie umieszczona w zbiorze

Wykonawszy wszystkie kroki przygotowujące możemy przejść do właściwej części rozwiązania:

1. Ustanawiamy zmienną . Zmienna jest inkrementowana, gdy komórka w zmienia wartość z 0 na 1 i dekrementowana przy zmianie z 1 na 0.
2. Sprawdzamy czy dla obecnych zawartości list w znajdują się podzbiory, dla których komórki w są wyzerowane. Jeśli nie, to przechodzimy do punktu 6.
3. Przenosimy liczbę z do , zerujemy komórki tablicy dla podzbiorów z i sprawdzamy, które z tych podzbiorów spełniają założenia zadania. Dla nich ustawiamy w komórkach wartość . Jeśli te wszystkie podzbiory to spełniają, to przechodzimy do punktu 6.
4. Sprawdzamy czy dla podzbiorów niespełniających założeń zadania, liczba jest ich ostatnim elementem. Jeśli nie, to przechodzimy do punktu 6.
5. Jeśli lista jest pusta, wtedy nie ma rozwiązania i taką odpowiedź zwracamy. W przeciwnym razie cofamy się do ostatniej liczby przerzuconej z listy do i przenosimy ją z powrotem do . Przypisujemy pod zmienną wartość tej . Sprawdzamy, które podzbiory z wciąż spełniają założenia zadania i odpowiednio modyfikujemy tablicę . Przechodzimy do punktu 3.
6. Sprawdzamy czy . Jeśli tak, to kończymy pracę i zwracamy rozwiązanie.
7. . Jeśli , wtedy nie ma rozwiązania i taką odpowiedź zwracamy. Jeśli nie, to przechodzimy do punktu 1.
8. **Pseudokod**



1. **Koncepcja generacji danych**

Tworzenie danych losowych do badań złożoności czasowej oraz poprawności algorytmu odbywa się w prosty sposób. Przed rozpoczęciem generacji użytkownik podaje jak duże podzbiory chciałby otrzymać – robi to poprzez przekazanie do programu wartości mówiącej jaką częścią całego zbioru może być maksymalnie podzbiór. Następnie ze zbioru wybieramy losową liczbę elementów

(z przedziału ) i dodajemy tak otrzymany podzbiór do listy naszych podzbiorów. Zastosowanie tego systemu nie daje nam gwarancji otrzymania zestawu, który da się podzielić, co z jednej strony może znacząco wydłużyć czas pracy algorytmu, ale z drugiej daje bardziej wiarygodny obraz mierzenia złożoności.

1. **Złożoność obliczeniowa**

Złożoność zaproponowanego przeze mnie algorytmu w najgorszym możliwym przypadku to , gdzie Wynika to z faktu, że przy pesymistycznym układzie podzbiorów, algorytm tak czy tak będzie musiał sprawdzić wszystkie możliwe kombinacje.

Jednakże przeprowadzone testy wskazują, że złożoność, jest o wiele mniejsza i można ją przybliżyć wzorem , gdzie . Tak zdecydowana różnica jest efektem tego, że zazwyczaj algorytm nie musi sprawdzać wszystkich możliwych kombinacji, a jedynie ich część, ewentualnie w najbardziej optymistycznym przypadku rozwiązanie może być znalezione nawet po jednokrotnym przejściu przez zbiór elementów.