אלגוריתם מתקדם - מטלה 2

מאור אופק

אלעזר פיין

1. ניתן להשתמש באלגוריתם שמשתמש באלגוריתם שלמדנו בכיתה (DP) ע"י שינוי הערכים במערך: (הוספת מספר גדול M מספיק לכל הערכים, הוספה שלו N פעמים למערך וחיפוש אחר סכום חדש N*M + SUM). אלגוריתם זה זהה בסיבוכיות (התוספת היא O(n).

```java
public class SubsetSum_Q2 {
    static boolean isSubsetSumForAllNumbers(int[] set, int sum) {
        int bigNumber = (int) 1E6; // assume very big number → at least as big as abs(min(set))
        List<Integer> transformedSet = Arrays.stream(set).map(x → x + bigNumber).boxed().collect(Collectors.toList());
        int n = transformedSet.size();
        for (int i = 0; i < n; i++) {
            transformedSet.add(bigNumber);
        }
        return isSubsetSumForPositiveNumbers(
                transformedSet.stream().mapToInt(x → x).toArray(),
                transformedSet.size(),
                sum: n * bigNumber + sum);
    }

    static boolean isSubsetSumForPositiveNumbers(int[] set, int n, int sum) {

        boolean[][] subset = new boolean[sum + 1][n + 1];

        // If sum is 0, then answer is true
        for (int i = 0; i ≤ n; i++)
            subset[0][i] = true;

        for (int i = 1; i ≤ sum; i++)
            subset[i][0] = false;

        for (int i = 1; i ≤ sum; i++) {
            for (int j = 1; j ≤ n; j++) {
                subset[i][j] = subset[i][j - 1];
                if (i ≥ set[j - 1]) subset[i][j] = subset[i][j] || subset[i - set[j - 1]][j - 1];
            }
        }
        return subset[sum][n];
    }

    public static void main(String[] args) {
        int[] A = {9, -3, 2, 4, -5, -1};
        int S = 7;
        System.out.println(isSubsetSumForAllNumbers(A, S));
    }
}
```

האלגוריתם עובד (אם קיים סכום לתת קבוצה אז גם קיים סכום עם SHIFT לתת קבוצה עם SHIFT), אך לא הצלחנו לשחזר בדרך זו את תת הקבוצה שמצריך סעיף ג', לכן השתמשנו באלגוריתם אחר בסוף:

<u>אלגוריתם ב'</u>

1. נגדיר את הקבוצות B כסכום האיברים השליליים של קבוצה A, ו C סכום האיברים החיוביים של קבוצה A.

2. אם קיים Ai==s שמור את האיבר ותחזיר TRUE

אחרת:

נגדיר FUNC(i,s) נבדוק אם קיימת תת קבוצה שסכומם S

3. עבור B > S > C מתקיים שלכל FUNC(I,S) יהיה FALSE

4. ניצור מטריצה בגודל n*(C-B) , ונמלא בעזרת רקורסיה:

FUNC(1,S)==(A1==S)

עבור i=2.....n FUNC(1,S)=FUNC(i-1,S) OR (Ai==S) OR FUNC(i-1,S-Ai)

למציאת תתי הקבוצות נמלא את הטבלה עם הדוגמא:

<u>שחזור תת קבוצה</u>

1.בעמודה S נעבור על כל האיברים עד שנמצא את ה TRUE הראשון בעמודה ונוסיף את איבר זה לרשימה, אם לא נמצא פשוט להחזיר FALSE (לא קיימת תת קבוצה).

2.נתחיל ב (i-1,s-Ai) אם הוא הראשון בעמודה נוסיף לתת קבוצה, אחרת נעבור על השורות עד שנמצא את הTRUE הראשון בעמודה.

3. אם Ai !=Si נחזור לשלב השני, אחרת נוסיף לתת קבוצה.

4.נחזיר את התת קבוצה.

סיבוכיות O(n*(C-B)) שזה זמן מילוי המטריצה.

Ai/Si	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
9	F	F	F	F	F	F	F	F	F	T	F	F	F	F	F	F	F	F	T	F	F	F	F	F	F
-3	F	F	F	F	F	F	T	F	F	T	F	F	F	F	F	T	F	F	T	F	F	F	F	F	F
2	F	F	F	F	F	F	T	F	T	T	F	T	F	F	F	T	F	T	T	F	T	F	F	F	F
4	F	F	F	F	F	F	T	F	T	T	T	T	T	T	F	T	F	T	T	T	T	T	T	F	T
-5	F	T	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	T
-1	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T

תת קבוצה:

-5, 4, 2, -3, 9,

.2

אלגוריתם:

```python
def count_prefixes_that_are_suffix(t, p, k):
    """
    There are only m possible prefixes of p
    so we just check for each of them if they are a suffix of t[0:k]
    :param t: The text.
    :param p: The pattern.
    :param k: Determines which prefix of t we investigate.
    :return: Count of prefixes of p that are a suffix of t[0:k]
    """
    count = 0
    n, m = len(t), len(p)
    for i in range(k - min(k, m), k):
        if p[0:k - i] == t[i:k]: # since it's python it automatically cuts off at len(arr) - ([1,2][0:100] = [1,2])
            count += 1
    return count
```

<u>סיבוכיות</u>: O(min(k, m) ^ 2)

<u>נכונות</u>:

קיימים רק m רישות ל p ואנחנו בודקים כל אחת אם היא סיפא של T[1..k]

לכן אם קיימת רישא לP שמקיימת את התנאי, אנחנו נמצא ונספור אותה.

.3

.א

state = 0, i = 1

δ (1, B) = δ (0, B) = 1

δ (1, C) = δ (0, C) = 0

$state$ = δ (0, P [2]) = δ (0, A) = 0

$state$ = 0, i = 2

δ (2, A) = δ (0, A) = 0

δ (2, C) = δ (0, C) = 0

$state$ = δ (0, P [3]) = δ (0, B) = 1

$state$ = 1, i = 3

δ (3, A) = δ (1, A) = 2

δ (3, C) = δ (1, C) = 0

$state$ = δ (1, P [4]) = δ (1, B) = 1

$state$ = 1, i = 4

δ (4, B) = δ (1, B) = 1

δ (4, C) = δ (1, C) = 0

$state$ = δ (1, P [5]) = δ (1, A) = 2

$state$ = 2, i = 5

δ (5, A) = δ (2, A) = 0

δ (5, C) = δ (2, C) = 0

$state$ = δ (2, P [6]) = δ (2, B) = 3

$state$ = 3, i = 6

δ (6, A) = δ (3, A) = 2

δ (6, B) = δ (3, B) = 4

$state$ = δ (3, P [7]) = δ (3, C) = 0

$state$ = 0, i = 7

δ (7, B) = δ (0, B) = 1

δ (7, C) = δ (0, C) = 0

$state$ = δ (0, P [8]) = δ (0, A) = 0

$state = 0, i = 8$

$\delta(8, A) = \delta(0, A) = 0$

$\delta(8, C) = \delta(0, C) = 0$

$state = \delta(0, P[9]) = \delta(0, B) = 1$

$state = 1, i = 9$

$\delta(9, A) = \delta(1, A) = 2$

$\delta(9, B) = \delta(1, B) = 1$

$\delta(9, C) = \delta(1, C) = 0$

טבלת מצבים:

states	a	b	c
0	0	1	0
1	2	1	0
2	0	3	0
3	2	4	0
4	5	1	0
5	0	6	0
6	2	4	7
7	8	1	0
8	0	9	0
9	2	1	0

ב.

T = “ABABABABBABCAA”

T	A	B	A	B	A	B	A	B	B	A	B	C	A	A
state	0	1	2	3	2	3	2	3	4	5	6	7	8	0

ג.

אוטומט

1	0	0	3	2	1	1	0	0

טבלת הפעולות

0	1	2	3	2	3	2	3	4	5	6	7	8	0

.4

א.

סיבוכיות התוכנית O(n) -

תוכנית זו יכולה לא למצוא מופע.

למשל עבור

P="abb"

T="ababb"

לא נמצא את מופע למרות שקיים.

I=1,

j=1

P[j]=T[i]

j!=m

i=2, j=2

i=2

j=2

P[j]=T[i]

j!=m

i=3, j=3

i=3

j=3

P[j]!=T[i]

i=4, j=1

i=4

j=1

P[j]!=T[i]

i=5, j=1

i=4

j=1

P[j]!=T[i]

i=6, j=1

ב.

סיבוכיות התוכנית O(mn) -

תוכנית זו עלולה לא למצוא אף מופע למרות שיש.

למשל עבור

P="abb"

T="ababb"

לא נמצא את מופע למרות שקיים.

i=1

j=1

P[j]=T[i]

j! =m

i=2, j=2

i=2

j=2

P[j]=T[i]

j! =m

i=3, j=3

i=3

j=3

P[j]! =T[i]

i=4, j=1

i=4

j=1

P[j]! =T[i]

i=5, j=1

i=4

j=1

P[j]! =T[i]

i=6, j=1

ג.

סיבוכיות O(NM)

פה ימצאו כל המופעים.

תוכנית זו תמצא את כל המופעים .

K זה הפוינטר על איפה נמצאים, אם נתקל בדפוס לא נכון נחזור למקום K+1 ונמשיך משם שוב(נאפס את J ו), משמע לא נדלג על שום מידע גם בעת נמצא דפוס נכון נחזור ל K+1 ונדע שמצאנו, אפשר לראות את זה על דוגמא (עשינו משהו ממש דומה בכיתה)

P= ”aba”

T=”ababa”