Z kwiatka na kwiatek

Wskazówka

Pasikonik startuje od początkowego kwiatka, możemy przyjąć, że ma on numer 0, jak początkowy znak w stringu opisującym stan kwiatków. Wprowadzamy pomocniczą zmienną o nazwie *pos* (od ang. *position*), która będzie oznaczać kolejną pozycję pasikonika. Za każdym razem staramy się skoczyć jak najdalej, jednak nie dalej, niż na ostatni kwiatek (stąd użycie funkcji `min()`). Takie „krótkowzroczne" podejście jest określane mianem *algorytmu zachłannego* (ang. *greedy*). Jeśli w wyniku takiego maksymalnego skoku wylądowalibyśmy na złamanym kwiatku, wtedy skracamy skok o 1 i próbujemy znowu, ewentualnie powtarzając do skutku skracanie skoku. Jeśli długość skoku zmaleje w ten sposób do zera, to znaczy, że ruch pasikonika jest niemożliwy i nie jest on w stanie dotrzeć do następnego (a więc i ostatniego) kwiatka.

Osiągnięcie przez zmienną *pos* wartości nie mniejszej niż $n - 1$ (numer ostatniego kwiatka, czyli znaku w ciągu) oznacza sukces pasikonika.

Kod w języku C++

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
  int n, d, ans = 0;
  string s;
  cin >> n >> d >> s;
  int pos = 0;
  while(pos < n - 1)
  {
    int i = min(n - 1, pos + d);
    while(s[i] == '0')
      i--;
    if(i == pos)
    {
      cout << -1 << endl;
      return 0;
    }
    pos = i;
    ans++;
  }
  cout << ans << endl;
}
```

Instrukcja `return 0` użyta po instrukcji warunkowej powoduje natychmiastowe zakończenie pracy programu.

Kod w języku Python

```
n, d = map(int, input().split())
s = input()
i = j = 0;
while i < n - 1 and j < n:
  if s[i] == '1':
    i += d
    j += 1
  else:
    i -= 1
if j >= n:
  print(-1)
else:
  print(j)
```

Uwagi

Gwoli ścisłości, instrukcja `return 0` powoduje zakończenie funkcji `main()`. Gdyby zależało nam na wymuszeniu zakończenia programu z wnętrza jakiejś innej funkcji, wtedy należy użyć instrukcji `exit(0)`.

Dla tego problemu algorytm zachłanny jest najprostszy, najefektywniejszy i – co najważniejsze – poprawny. Istnieją zagadnienia, dla których takie podejście daje nieoptymalny wynik, na przykład *problem plecakowy*, który należy rozwiązywać metodą programowania dynamicznego.

Bardziej zaawansowani uczniowie mogą spróbować rozwiązać ten problem przy pomocy wspomnianej metody programowania dynamicznego lub przeszukiwaniu grafu w głąb.