# Zadanie: NPN
# Najdłuższy podciąg niemalejący

**Warsztaty ILO, grupa olimpijska, dzień 1. Dostępna pamięć: 128 MB.**

## Rozwiązanie wzorcowe $O(n)$

Zadanie sprowadza się do znalezienie najdłuższego podciągu postaci $1^a2^b1^c2^d$, gdzie $c^n$ oznacza liczbę $c$ powtórzoną $n$ razy. $a, b, c$ oraz $d$ mogą być oczywiście równe 0. Łatwo zweryfikować, że zadanie się do tego sprowadza, ponieważ wystarczy przedział od pierwszej dwójki do ostatniej jedynki odwrócić i otrzymamy podciąg składający się najpierw z samych jedynek, a potem z samych dwójek (a zatem niemalejący). Aby znaleźć długość naszego najdłuższego ciągu, zastosujemy programowanie dynamiczne. Dla każdego prefiksu ciągu o długości $i$, policzymy sobie, jaki jest najdłuższy podciąg opisany wyżej, taki że:

- jeszcze nie dodaliśmy żadnej dwójki do podciągu (zatem mamy coś postaci $1^a$),
- już pojawiła się dwójka, ale nie pojawiła się kolejna jedynka (zatem coś postaci $1^a2^b$),
- pojawiła się kolejna jedynka po fragmencie dwójek ($1^a2^b1^c$),
- i ostatecznie pojawił się kolejny ciąg dwójek $1^a2^b1^c2^d$).

Jeśli ponumerujemy sobie te stany od 1 do 4, to wyniki dla naszych stanów i kolejnych prefiksów możemy obliczyć następująco:

Jeśli $i$-ta liczba to 1:

```
dp[i][1] = dp[i - 1][1] + 1; // bierzemy najdluzszy podciag konczacy sie co najwyzej
na pozycji i-1 zlozony z samych jedynek i doklejamy na koniec cyfre 1, poniewaz taka
napotkalismy.
dp[i][2] = dp[i - 1][2]; // bierzemy najdluzszy podciag 1^a2^b, ale nie doklejamy
naszej cyfry, poniewaz przeszlibysmy do stanu 3
dp[i][3] = max(dp[i - 1][3], dp[i - 1][2]) + 1; // dwie opcje: albo bierzemy wynik
wyzej, ale tym razem doklejamy cyfre, przechodzac do stanu 3, albo bierzemy wynik
dla mniejszego prefiksu w stanie 3 i wydluzamy go.
dp[i][4] = dp[i - 1][4]; // Skoro nasza cyfra to 1, to w stanie 4 juz nie mozemy
jej dodac.
```

a jeśli $i$-ta liczba to 2 (tym razem bez komentarzy, bo jest analogicznie):

```
dp[i][1] = dp[i - 1][1];
dp[i][2] = max(dp[i - 1][1], dp[i - 1][2]) + 1;
dp[i][3] = dp[i - 1][3];
dp[i][4] = max(dp[i - 1][3], dp[i - 1][4]) + 1;
```

Wynikiem zadania jest $max(dp[n][1], dp[n][2], dp[n][3], dp[n][4])$.