Wyszukiwanie z wildcardami w tekście i wzorcu

Kamil Rajtar

Czerwiec 2020

Opis problemu

Na wejściu do algorytmu dostajemy tekst i wzorzec w których oprócz liter moga wystepować wildcardy (?) pasujace do dowolnego znaku. Na wyjściu ma sie znaleźć lista pozycji w których wzorzec pasuje do tekstu.

Wersja bez wildcardów

W rozwiazaniu problemu użyty jest splot obliczany za pomoca szybkiej transformaty Fouriera (FFT) zdefiniowany nastepujaco:

$$p \otimes t \stackrel{\text{def}}{=} (\sum_{j=0}^{m-1} p_j t_{i+j}, 0 \le i \le n-m)$$

Normalne (bez wildcardów) dopasowanie wzorca p do tekstu t na pozycjach od i do i+m możemy obliczyć za pomoca splotu. Korzystamy z własności wzoru:

$$\sum_{j=0}^{m-1} (p_j - t_{i+j})^2 = \sum_{j=0}^{m-1} (p_j^2 - 2p_j t_{i+j} + t_{i+j}^2)$$

Zobaczmy że lewa cześć równania równa 0 to znaleźliśmy dopasowanie. Tekst nie różni sie od wzorca na żadnej z m kolejnych pozycji. Prawa strone potrafimy szybko obliczyć ponieważ podniesienie każdej pozycji do kwadratu wykonywane jest w czasie stałym a środkowy składnik liczmy za pomoca FFT.

Wersja z wildcardami

Zdefiniujmy ciagi:

$$p'_j = \begin{cases} 0 & \text{p}_j = '?' \\ 1 & \text{wpp} \end{cases}$$

$$t'_j = \begin{cases} 0 & \text{t}_j = '?' \\ 1 & \text{wpp} \end{cases}$$

Wtedy łatwo jest widać że nastepujace równanie jest naturalnym rozszerzeniem rozwiazania poprzedniego problemu.

$$\sum_{j=0}^{m-1} p'_j t'_{i+j} (p_j - t_{i+j})^2 = 0$$

Zobaczmy że tekst cześć pod suma jest zawsze nieujemna p'_j oraz t'_{i+j} przyjmuja wartości $\{0{,}1\}$ oraz kwadrat różnicy jest zawsze nieujemny. Wiec zastanówmy sie kiedy suma jest równa zero. Wtedy kiedy wszystkie składniki sa równe zero. Kiedy składnik jest równy zero? W jednym z 3 przypadków. $p'_j = 0$ wtedy mamy wildcard we wzorcu, $t'_{i+j} = 0$ wtedy mamy wildcard w tekście, $(p_j - t_{i+j})^2$ wtedy litery wzorca i tekstu sa równe. Zobaczmy że to zachodzi wtedy i tylko wtedy gdy mamy do czynienia z dopasowaniem.

Po rozpisaniu nawiasu dostajemy forme:

$$\sum_{j=0}^{m-1} (p'_j p_j^2 t'_{i+j} - 2p'_j p_j t_{i+j} t'_{i+j} + p'_j t_{i+j}^2 t'_{i+j})$$

Takie rozwiniecie łatwo jest obliczyć wykorzystujac 3xFFT oraz podstawowe operacje na ciagach.

Złożoność

Trywialnie jest pokazać że algorytm działa $O(n \log n)$ gdy użyjemy FFT dla tekstu i patternu rozszerzonego do długości tekstu. Można jednak osiagnać $O(n \log m)$ gdy podzielimy tekst na n/m zachodzacych na siebie kawałków wielkości $2m$ i dla każdego policzymy matching ze wzorcem a nastepnie wyniki złaczymy. Zobaczmy że jest to poprawne ponieważ dla każdego miejsca poczatkowego istnieje kawałek, który zawiera cześć od długości co najmniej m od tego miejsca. Złożoność: $n/m * O(m \log m) = O(n \log m)$.