АНАЛИЗ на XOR

Наивното решение с квадратична сложност O((*b*–*a*+1–*n*) × *n*), което проверява всяка *n*-орка, преизчислявайки операцията отначало, не трябва да получи точки. То може лесно да се преобразува в линейно O(*b*–*a*), като се ползва, например, очевидното . За поне такова решение са предвидени 20% от точките.

По-нататъшното ускоряване на изчислителния процес предполага други методи или по-задълбочени наблюдения.

За изчисляването на „побитово изключващо или“ върху *n*-орка от последователни числа е достатъчно операцията да се извърши с не повече от три от последните елементи. Може да се забележи, че при четно начало на *n*-орката всеки четвърти междинен резултат е нула, а при нечетно – всеки четвърти след първия е равен на първия. Това просто наблюдение прави сложността на предишното решение O(*b*-*a*-*n*), което е съществено при сравними *n* и *b*-*a*. Доказателството на тези свойства е елементарно и се основава на добре известния факт (който следва пряко от дефиницията), че, всъщност, редът на изпълнението на операцията „побитово изключващо или“ не е от значение – важна за резултата е само четността на броя единици във всички операнди във всеки от двоичните разреди. Тогава *a*⊕(*a*+1)⊕(*a*+2)⊕(*a*+3) = (*a*⊕(*a*+1))⊕((*a*+2)⊕(*a*+3)). Ако *a* е четно, *a*+1 се различава от него само по най-младшия бит, т. е. *a*⊕(*a*+1) = 1, аналогично (*a*+2)⊕(*a*+3) = 1 и, следователно, (*a*⊕(*a*+1))⊕((*a*+2)⊕(*a*+3)) = 1⊕1=0. Ако *a* е нечетно, *a*+1 е четно и оттам *a*⊕(*a*+1)⊕(*a*+2)⊕(*a*+3)⊕(*a*+4) = *a*⊕((*a*+1)⊕(*a*+2)⊕(*a*+3)⊕(*a*+4)) = *a*⊕0 = *а*.

По-нататъшно следствие е, че ако *n* е нечетно, резултатите върху *n*-орките с начала *a*, *a*+2, *a*+4, … растат (с по 2 на всяка стъпка, което, всъщност, не е важно за нашия алгоритъм), както и тези, с начала *a*+1, *a*+3, *a*+5, …. Тогава за получаване на резултата в този случай е достатъчно да изведем по-големия резултат от операцията върху последните две *n*-орки в интервала. При описаното бързо пресмятане на операцията, това решение е константно.

Остава да разгледаме най-интересния случай на четно *n*. При него търсим *n*-орка, в която колкото може по-старши битове се срещат нечетен брой пъти. Всъщност, от казаното по-горе е ясно, че, щом *n* е четно, като разгледаме пак операцията върху последователни двойки числа, ще получим следното:

* Ако *n* се дели на 4 и началото *s* е четно, резултатът е 0;
* Ако *n* не се дели на 4 и началото *s* е четно, резултатът е 1;
* Ако началото *s* е нечетно, резултатът е:
* при *n*=4*k*+2: *r*=*s*⊕[((*s*+1)⊕(s+2))⊕((*s*+3)⊕(s+4))⊕… ⊕((*s*+*n*-3)⊕(*s*+*n*–2))]⊕(*s*+*n*–1). Както показахме, изразът в средните скоби е 0, следователно в този случай се получава *r*=*s*⊕(*s*+*n*–1).
* при *n*=4*k*: *r*1=*s*⊕[((*s*+1)⊕(s+2))⊕((*s*+3)⊕(s+4))⊕…]⊕[(*s*+*n*–3)⊕(*s*+*n*–2)]⊕(*s*+*n*–1). Изразът в първите средни скоби е нула, а във вторите е 1. Така в този случай *r*1=*s*⊕(*s*+*n*‑1)⊕1=*r*⊕1. Формулата е същата, както в предишния случай, с най-младши бит нула.

Казано в резюме, няма смисъл да разглеждаме *n*-орки с четно начало. Търсим такова нечетно *s*, *a* ≤ *s* ≤ *b*–*n*+1, което максимизира съответното *r*.

*a*⊕*b* е число, най-старшият значещ бит на което е възможният (и осъществим!) най-значещ бит и на *r*. Осъществяването му в резултата води до следното: ако това е бит номер *v* (броено от 0 от дясно наляво), в интервала [*a*, *b*] задължително се среща числото *t*, чийто двоичен запис започва с общото начало на *a* и *b* (ако има такова) и завършващо с . Тогава *s* е някъде в околността на [*t*-1, *t*]. Можем са положим *a* =max(*a*, *t*–*n*+1) и *b* =min(*b*, *t*+*n*–1), което прави търсената област не по-голяма от 2*n* (което значи нечетните начала в нея – не повече от *n*). Линейния алгоритъм с начало (новото) *a* и брой стъпки не повече от *n* решава въпроса в този случай.