Анализ

Формално условие: Започваме от някое число в редицата(curr=a_t) и трябва да вземем на всяка стъпка такова a_j, което има общ бит 1 с curr и да сложим битове 1 в curr там където в a_j ги има а в curr са 0. Целим след изпълнението на всички стъпки да имаме число в curr с максимален възможен брой битове 1.

В анализа чрез len означавам максималната дължина на число в двоична бройна система, а с B максималната в десетична бройна система.

Основна идея

Цялото решение стъпва на две наблюдения:

1) curr е ИЛИ на всички досега взети числа. Затова a_j може да се вземе, ако има общ бит с което и да е от взетите числа, а не само с последното.

2) Ако свържем всеки две числа с общ бит 1, от всяко число се стига до всички числа от неговата компонента (в BFS ред всяко следващо е свързано с някое вече взето) и не може да се излезе извън нея. Следователно отговорът е максималният брой различни битове 1 в една компонента (0, ако всички числа са 0).

Подзадача №0

Както винаги това е подзадача с примерните тестове за обратна връзка от системата.

Подзадача №1

Щом $a_i<2^2$, това означава че имаме четири вида числа(0, 1, 2, 3) и съответно три случая:

- Ако имаме тройка -> Отговорът е 2, защото $3=11_{(2)}$.
- Ако имаме само нули -> Отговорът е 0.
- Във всички други случаи -> Отговорът е 1. ($1=01_{(2)}$ и $2=10_{(2)}$ нямат общ бит, 1&2=0, затова не могат да се комбинират)

Постигната сложност: $O(N)$

Подзадача №2

В тази подзадача можем да правим всякакъв вид brute force понеже n≤10. Пробваме всяко начално число и всяка подредба на останалите числа, симулираме играта и пазим най-добрия брой единици.

Постигната сложност: $O(N!)$

Подзадача №3

Тук N е малко по-голямо от подзадача 2. Това че е до 19 ни подсказва, че можем да правим dp по маски със следния стейт:

dp[mask]

Където mask е маска на взетите числа (бит i е 1, ако числото a_i е взето), а dp[mask] е истина, ако това множество може да бъде взето в някакъв ред. curr се определя само от маската (то е ИЛИ на

взетите числа) и може да се предизчисли за всички маски. От mask можем да добавим число i, което още не е взето, ако a_i&curr>0. Началните състояния са маските с едно число, а отговорът е най-големият брой единици в curr по всички достижими маски.

Постигната сложност: $O(N2^N)$

Подзадача №4

Тази подзадача я направих за по-бавни реализации на следващата.

Постигната сложност: $O(N^3)$

Подзадача №5

В тази подзадача вече започваме да навлизаме в идеята за сто точки. Ще построим непретеглен граф, в който върховете са индекси от редицата и във всеки връх имаме числото от съответната позиция. Тогава ще имаме ребро между два върха само когато ако сме взели едното число можем да вземем другото или условно казано трябва (a_i&a_j)≥1. След това ясно се вижда, че оптималния отговор обхваща компонента от този граф. Следователно трябва да разбием на компоненти и да намерим колко различни бита 1 се срещат във всяка. Отговорът е максималното от някоя група. Проверката (a_i&a_j) е една операция, тъй като $a_i<2^{20}$.

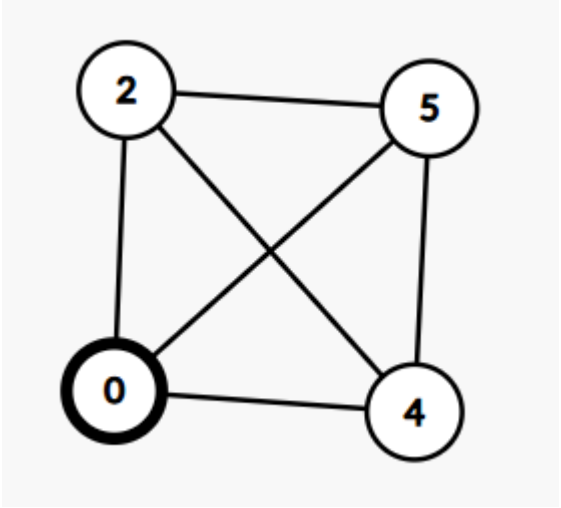
Постигната сложност: $O(N^2+N\text{len})$

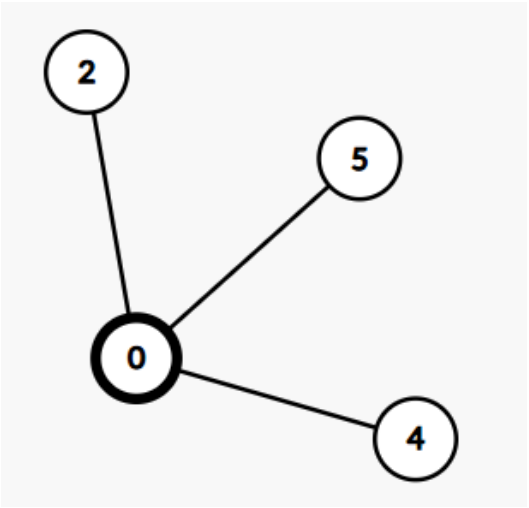
Подзадача №6

Сега трябва да подобрим идеята от 5 подзадача. Проблемът в нея е че имаме прекалено много ребра. Ще използваме известен подход за решаване на този проблем, а именно да добавим нови върхове, които ще са съответно битовете единици. Съответно непретегленият граф ще е от върхове числа и върхове битове. Ребрата в него ще са от число към бит само ако този бит е 1 в числото. По този начин отново отговорът е в една компонента. Разликата обаче е че имаме доста по-малко ребра.

Има и още един по-добър начин за представяне. Можем да махнем всички върхове числа и да останат само върхове битове. Ребрата ще са между битовете 1 в някое число, като за да нямаме len^2 ребра ще са само от битовете 1 в число към най-младшия бит 1 в същото число, понеже така пак са в една компонента. Да го направим с пример:

Имаме числото $53=110101_{(2)}$.





Можем да забележим че и в двата случая четирите върха са в една компонента. Съответно отново отговорът ще е максималния брой на върхове в една компонента. Внимание: броят се само битове, които се срещат в поне едно число (и само върховете битове, а не върховете числа). Иначе битовете, които не се срещат никъде, са отделни компоненти с размер 1 и при вход само от нули ще получим 1 вместо 0.

Постигната сложност: $O(N\text{len})$

Подзадача №7

В следващите подзадачи започва забавната част :). В тази специално единственото нещо което трябва да направим е да ползваме дълги числа за преобразуване на числата в двоична бройна система. Това става като делим десетичния низ на 2, както на ръка: минаваме от най-старшата към най-младшата цифра и пазим остатък r. За цифра d взимаме cur=r·10+d, новата цифра на частното е cur/2, а остатъкът cur mod 2. Остатъкът след последната цифра е следващият бит. Едно деление е O(B), а са нужни len деления.

Постигната сложност: $O(N\text{Blen})$

Подзадача №8

Нека видим приликите между число представено в двоична и в осмична бройна система с примерно число 98 = $1100010_{(2)}$ = $142_{(8)}$. Можем да забележим, че всяка цифра от осмичната бройна система отговаря на три бита от двоичното представяне(отбелязани са с цветове). Трите бита отговарят на двоичното представяне на съответната осмична цифра и това става защото 8 = 2^3. Следователно ако представим числото в 2^x съответната „цифра“(цифрата може да е и по-голяма от 9) ще отговаря за x бита в двоична бройна система. В тази подзадача ще precompute-нем всички числа от 1 до 2^6-1. По този начин можем по-бързо да представим числото в двоична бройна система. С други думи, делим десетичния низ на 2^6=64 вместо на 2. Остатъкът (от 0 до 63) директно дава 6 бита, като r&1, (r>>1)&1 и т.н., а преминаванията стават 6 пъти по-малко.

Постигната сложност: $O(N{\cdot}B{\cdot}\text{len}/6)$

Подзадача №9

В тази подзадача не ни трябва целият двоичен запис, а само позициите на единиците. Затова вместо на 2 или на 2^6 делим десетичния низ на 2^{32}. Остатъкът от всяко деление е число от 0 до $2^{32}-1$, т.е. точно 32 бита от двоичния запис: първият остатък дава битовете [0, 31], вторият – [32, 63] и т.н. Позициите на единиците във всяка такава дума намираме с побитови операции, без да строим двоичното число.

Едно деление правим както в предишните подзадачи. Минаваме по цифрите на низа от най-старшата към най-младшата и пазим остатък r. За цифра d изчисляваме cur = r·10 + d. Новата цифра на частното е cur / 2^{32}, а новият остатък е cur mod 2^{32}. Тъй като r < 2^{32}, имаме cur < 10·2^{32}, т.е. стойността се побира в 64-битов тип. След последната цифра остатъкът е следващата 32-битова дума. Повтаряме, докато частното стане 0, което са най-много len/32 деления.

Постигната сложност: $O(N \cdot B \cdot \text{len}/32)$

Подзадача №10

Горната реализация все още обработва по една десетична цифра на стъпка. Можем да обработваме по 9 цифри наведнъж, т.е. да гледаме числото в основа 10^9. Десетичният низ разбиваме на блокове по 9 цифри, започвайки отдясно (най-старшият блок може да е по-къс). Така вместо B цифри имаме B/9 блока, всеки със стойност под 10^9.

Делението на 2^{32} е същото като в подзадача 9, но върху блокове вместо цифри: за блок b изчисляваме cur = r·10^9 + b. Новият блок на частното е cur / 2^{32}, а новият остатък е cur mod 2^{32}.

Нужни са две проверки, че това е коректно. Първо, cur ≤ (2^{32} − 1)·10^9 + (10^9 − 1) = 2^{32}·10^9 − 1, което е около 4,3·10^{18} и е по-малко от 2^{63}, така че се побира в 64-битов тип. Второ, от cur < 2^{32}·10^9 следва, че cur / 2^{32} < 10^9, т.е. новият блок на частното също е валиден блок и частното остава записано в основа 10^9.

Пример с числото 67

Нека вече сме обработили цифрата 6. В двоичен вид тя е $110_{(2)}$. Сега добавяме цифрата 7. Трябва да изчислим:

$$R = (110_{(2)} \times 10) + 7$$

Вместо да умножаваме целите числа, ние преминаваме през всеки бит на $110_{(2)}$ отдясно наляво (от най-младшия към най-старшия) и прилагаме преноса (carry):

1. **Бит 0 (стойност 0)**
   - Изчисляваме 0 × 10 + 7 = 7.
   - Новият най-младши бит е 7 mod 2 = 1.
   - Преносът към следващия бит е $\lfloor 7 / 2 \rfloor$ = 3.
2. **Бит 1 (стойност 1)**
   - Изчисляваме 1 × 10 + 3 = 13.
   - Новият бит е 13 mod 2 = 1.
   - Преносът към следващия бит е $\lfloor 13 / 2 \rfloor$ = 6.
3. **Бит 2 (стойност 1)**

- Изчисляваме 1 × 10 + 6 = 16.
- Новият бит е 16 mod 2 = 0.
- Преносът към следващия бит е $\lfloor 16 / 2 \rfloor$ = 8.

4. **Продължаваме с преноса (8)**

- 8 → бит 0, пренос 4.
- 4 → бит 0, пренос 2.
- 2 → бит 0, пренос 1.
- 1 → бит 1, пренос 0.

Резултатът от битовете (от най-младшия): 1, 1, 0, 0, 0, 0, 1, което обърнато е $1000011_{(2)}$ (64 + 2 + 1 = 67).

Постигната сложност: $O(N \times (len/32) \times (B/9))$

Автор: Никола Манолов
Идея за последните подзадачи: Румен Михов