1. Використати хеші. Захешувати можна все, лише потрібно визначити правила ізоморфізму множин.

2. Розглянути задачу в теорії графів.

3. Бінарний та тернарний пошуки - коли маємо в кінці відрізок [l;r] давайте переберемо відповідь на відрізку [l-T;r+T] (T приблизно залежить від ТЛ там вхідних даних, можна ставити приблизно Т = 20)

4. Відсікання по часу(наприклад ваше рішення працює занадто довго, давайте спробуємо просто вичерпавши весь тайм ліміт вивести ту відповідь, яку вважає на даному етапі самою вірною; щоб робити відсікання по часу можна використовувати clock()

http://cppstudio.com/post/561/ , або ж дві змінні - одна - кількість операцій, друга - максимальна кількість операцій, при котрій якщо перша змінна вже досягнула другу ми виведимо нашу відповідь)

5.Знаходження порядку вхідних даних за допомогою відправок та бін.пошуку по відповіді, викорстовуючи операцію assert

6. Швидке зчитування(scanf/printf)

char ch;

inline void read(int &x){

x = 0;

while(1){

ch = getchar();

if(!(ch >= '0' && ch <= '9') return;

x \*= 10;

x += ch - '0';

}

}

Відповідно дану процедуру/метод треба модифікувати, якщо в нас є наприклад від'ємні числа.

7. Розглянути задачу з кінця(почати розв'язувати обернену задачу чи почати з кінця вхідних даних)

8.Якщо задача на бітові операції:

а) Можна розглянути біти незалежно один від одного, якщо це можна.

б) Використати бітовий бор.

9. Розписати задачу на формули( використати часткові суми)

10. Числа Фібоначі(швидке знаходження через матриці чи рекурсію з запам'ятовуванням(мемоізацією) )

11. Розгянуте щось в задачі як бітове подання(в системі числення 2)

12. Відсортувати/пошафлити вхідні дані, якщо це можливо.

13. Розбити задачу на sqrt блоків.

14. Розбити задачу на складні та прості запити.

15. Купа(велика кількість) різних компараторів в жадібних алгоритмах. Асимптотика вашого рішення тоді буде O(CntCmp \* n \* log n \* TIME\_PROCESS\_GREEDY)

16. Розібрати якийсь частковий випадок і почати тепер розв'язувати стартову задачу.

17. Якщо у вас є рішення "правильне", котре не заходе якийсь блок, і є рішення "в лоб"(примітивне , яке вірно працює, упевневшись при здачі на контесті), то можна написати генератор впидакових вхідних даних, які задовольняють обмеженням задачі, і перевірити відповіді цих двух рішень. Відповідно де не співпадає відповідь - ми знайшли тест, на котрому наше "правильне" рішення видає неправильну відповідь на тест.

18. Решето Ератосфена за O(n) дає змогу:

а)Можна порахувати i = 1,2,..n phi(i) за O(n)

б)Коли ми побудували масив lp[i] (мінімальний простий в рокзладі числа і), то ми можемо за O(log n) факторизувати наше число, на відміну від O(sqrt(n)) якби це робили просто "в лоб"

19. Вершиною в графі може бути що завгодно(число, масив, vector, map, etc)

20. Станом в динаміці може бути що завгодно(число, масив, vector, map, etc)

21. Використайте лист паперу та ручку/олівець.

22. Контролюйте свої емоції під час написання контесту. Не панікуйте. Спробуйте використати тайм-менеджмент під час написання змагання.

23. Якщо розумієте що до кінця контесту можете не написати рішення на повний бал, дебаживши його занадто довго, зашліть хоча б окремі блоки, щоб встигнути отримати хоча б якісь бали, а потім вже повернутися до вашого рішення на повний бал.

24. Використовуйте STL по максимуму(є вбудовані структури даних, котрі частково схожі на декартове дерево, і можна використати його при можливості).

25. Якщо розумієте що задача складна для вас - спробуйте проаналізувати, подивившись на ваших опонентів/однокімнатників по написанню контеста/в таблицю результатів, чи дійсно вона складна, чи можливо ви просто в іншу сторону від правильного рішення.

26. Динамічне програмування на дереві - це просто.

27. Подумайте як автор : який гірший випадок може зустрітися серед тестів - тест, котрий може повалити ваше рішення, можливо згенерувати занадто складно, чи придумати.

28. Якщо у вас два рішення, котрі якщо охайно об'єднати - буде повне рішення, чи зайде більша кількість тестів - об'єднуйте.

29. Пам'ятатйте : у вас може бути обмеження на кількість відправок. Якщо кількість обмежена, не поспішайте відправляти ваш розв'язок, краще 2-3 хв протестуйте його на тестах, придуманих вами ж.

30. Якщо ви намагаєтесь здати задачу жадібними алгоритмами, то спочатку доведіть що дп там не пройде. Якщо у вас не вийшло - пишіть динаміку.

31. якщо в задачі порахувати щось на відрізку , спробуйте звести цю задачу до двух запитів вигляду f(r) - f(l - 1), f(r) - відповідь від 1 до r

32. sparse table - побудувавши sparse table, то щоб знайти відповідь на відрізку l r , можна розгялнути r - l + 1 в системі числення 2, то можна пригати по нашому відрізку саме по тим степеням двійки, котрі є в розкладі довжини нашого відрізка, на котрому ми шукаємо відповідь)

33. Використовуйте структури даних щоб зменшити кількість операцій, котрі відпрацьовує ваше рішення.

34. Тайм ліміт може бути і через маленьку кількість елементів у масиві.

35. Аналізуйте тайм ліміт , меморі ліміт та обмеження в задачі - зазвичай на Всеукрі дають занадто малий запас/різницю очікувану від авторського рішення. Це може дати змогу дізнатися, яка там може бути асимптотика, відповідно можна тоді спробувати здогадатися яке рішення писав автор і мав на увазі для розв'язування даної задачі.

36. Намагайтесь заслати те,що по ідеї не повинно пройти, і у вас уже немає інших ідей як задачу можна вирішити. Якщо у вас залишилась одна задача на контесті і є час - пишіть всі ідеї, котрі вам приходять в голову.

37. <http://codeforces.com/blog/entry/48417>

Ідеї вирішування задач:

1.Графи(бфс, дфс, дейкстра, форд-белман, флойд, мін. каркас(крускал, прім, буровки), звести до дерева граф, макс. поток, LCA, мости, точки зчеплення)

2.ДП(бітмаски, пропихнути(вперед->),(назад<-),дп на відрізках, дп по профілю)

3.Рекурсія(перебір з поверненням). Мемоізація.

4.Стуктури даних(Дерево відрізків, корнева декомпозиція(розбити запити на корінь блоків), Фенвік, sparse table, DSU)

5.Геометрія(звести задачу до примітивів). Формула Піка - A = i + b / 2 - 1

6.Бінарний та тернарний пошуки.

7.Жадібні алгоритми.

8.Записати всі ідеї по задачі на аркуш паперу.

9.Аналіз обмеження в задачі(ТЛ,МЛ, обмеження по вхідним даним) - це може дати який час виконання автор очікує щоб отримати фулл бал по задачі.

10.Алго на рядках(префікс-,з-функція,хешування,бор, можливо ахо-корасік, Манакер, суф. масив)hashset,hashmap(unordered\_map,unordered\_set)

11.Рекурсивно залишити на стек відкладені операції.

12.Персистентність(Дерево відрізків, стек, черга).

13.Мінімум на черзі та на стеці за О(n).

14.Звести задачу до арфиметичної або геометричної прогресії.

15.Функція Ейлера.

16.Комбінаторика(C(n,k) = C(n-1,k-1) + C(n-1,k), C(n,k) = (n!) / ( (n - k)! \* k!) )

module - prime, C(n,k) % module = n! % module \* (n-k)! ^ (module - 2) \* k! ^ (module - 2) - Обернений елемент в кільці.

17.Решето Ератосфена за О(n log n log n) , O(n). Факторизація за log N за допомогою лінійого решета.

18.Бінарне піднесення до степені.

19.Числа Фібоначі та їх властивості.

20.НСД та НСК.

21.Використати рандом(рандомно перетасувати вхідні дані та вирішити задачу).

22.Відсікання по часу(кількості зроблених операцій).

23.Цифровий,бітовий бори.

24.Скан лайн.

25.Рішення в онлайні/оффлайні - якщо можна простіше вирішити задачу, спочатку зчитавши всі дані, а потім почати обробку з кінця(тобто офлайн спосіб).

26.Множення матриць. Піднесення матриці в степінь за допомогою бінарного піднесення до степеня. Використання матриці перехоодів при підрахунку динаміки.

27.Розподіляй та володарюй(задачі : сортування злиттям, к-сть інверсій).

28.Meet in the middle.

29.Теорія чисел.