1. Идея

При използване на техниката „разделяй и владей“ обикновено разделяме множество елементи на части, решаваме задачата за всяка част и накрая „сливаме“ отделните части. В повечето случаи разделяме интервала *[l;r]* на *[l;m]* и *[m+1;r]*, като двете половини са независими една от друга. Техниката CDQ се използва, когато елементите от едната част оказват влияние върху другата. Алгоритъмът за решаване на задачи чрез CDQ най-често изглежда така:

1. Разделяме *[l;r]* на *[l;m]* и *[m+1;r]*
2. Решаваме задачата за *[l;m]* и *[m+1;r]* поотделно
3. Пресмятаме „влиянието“ на *[l;m]* върху *[m+1;r]*
4. Сливаме *[l;m]* и *[m+1;r]*  
   *(Понякога трябва да решим задачата за [m+1;r], след като сме изчислили влиянието на [l;m] върху [m+1;r])*

Пример: example.cpp

1. Кога можем да използваме CDQ?  
     
   CDQ се използва най-често в задачи от типа “Insert/Query”, т.е. трябва да извършваме промени и да отговаряме на въпроси например върху масив или някаква структура от данни. Важни *(не абсолютно задължителни, но много често необходими)* условия за прилагане на техниката са:

* Знаем предварително какви промени ще извършваме и на какви въпроси ще отговаряме
* Бихме могли да решим задачата лесно, ако всички промени се извършват преди всички въпроси
* Заявките се „натрупват“ – например, при промените увеличаваме стойността на елемент с дадено число, а при въпросите питаме за сума. Други примери са заявки за минимум, максимум, НОД, НОК, побитовите операции AND, OR и XOR.

1. Приложения

Техниката може да се приложи върху задачи от по-горе описания вид или такива, които могат да се сведат до тях.

Често се използва за „сваляне“ на измерение в задачи, в които иначе бихме използвали многомерна структура от данни (например можем да използваме CDQ и нормално сегментно дърво вместо двумерно такова). Това е важно, тъй като многомерните структури обикновено използват много памет и имат голяма константа, докато CDQ ни позволява да използваме малко памет, като работи със същата сложност, но „скритата константа“ е малка и се държи доста по-добре. Освен това кодът става по-лесен за писане и съответно шансът за грешки е по-малък.

CDQ може да се използва в задачи с динамично програмиране, като изчисляването на state представим като заявка за въпрос, а след това изчислената стойност – като заявка за промяна.

В някои случаи техниката позволява да не имплементираме „динамична“ структура от данни и да постигнем същата цел със същата сложност (например вместо Dynamic CHT можем да използваме CDQ и CHT).

**Задачи:**

<https://vjudge.net/problem/UVALive-5871>

<https://zerojudge.tw/ShowProblem?problemid=c571>

<https://arena.infosbg.com/#/catalog/578/problem/101434> (IATI 2020 B13\_disorder)

<https://codeforces.com/contest/1093/problem/E>

CodeIT 2023/2024 Round 6 – company

<https://www.luogu.com.cn/problem/P2487>

<https://arena.infosbg.com/#/catalog/709/problem/101798> (ЛТИ 2022 B1\_otbor)

<https://tioj.ck.tp.edu.tw/problems/1921> (тагове: DP, Convex Hull Trick)

<https://arena.infosbg.com/#/catalog/643/problem/101617> (IATI 2021 A13\_news)

**Допълнителни материали:**

<https://robert1003.github.io/2020/01/31/cdq-divide-and-conquer.html>

<https://assets.hkoi.org/training2018/dc.pdf>

<https://assets.hkoi.org/training2021/adc.pdf>

<https://oi-wiki.org/misc/cdq-divide/>

<https://youkn0wwho.academy/topic-list/cdq_divide_and_conquer>