ИЗПИТ ПО "ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ" — СУ, ФМИ, 03. 09. 2018 Г.
(СПЕЦИАЛНОСТ "КОМПЮТЪРНИ НАУКИ", 1. ПОТОК, И ИЗБИРАЕМИЯ КУРС)

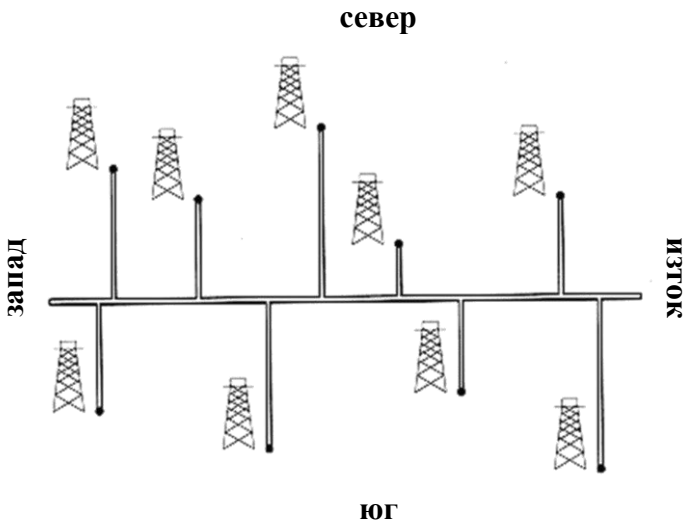
Задача 1. Намерете времевата сложност $T(n)$ на алгоритъма StupidSort в най-лошия случай.

Кой алгоритъм е по-бърз асимптотично — сортирането чрез вмъкване (InsertionSort) или StupidSort? Отговорът да се обоснове с помощта на подробно обяснено сравнение на времената на двата алгоритъма.

```
StupidSort(A[1...n])
if n = 2 and A[1] > A[2]
    swap(A[1],A[2])
else if n > 2
    m = ⌈2n/3⌉
    StupidSort(A[1...m])
    StupidSort(A[n-m+1...n])
    StupidSort(A[1...m])
```

Задача 2. Компания за добив на нефт планира да прекара дълъг нефтопровод в посока изток-запад през нефтено поле с n кладенеца. Всеки кладенец ще бъде свързан с нефтопровода чрез възможно най-къса тръба (т.е. в посока север-юг, както е показано на схемата). Търси се такова местоположение на нефтопровода, че сборът от дължините на тръбите, които го свързват с нефтените кладенци, да бъде възможно най-малък.

Предложете алгоритъм, който намира търсеното най-добро местоположение на нефтопровода за време $T(n) = O(n)$ в най-лошия случай.



Входните данни са два числови масива `X[1...n]` и `Y[1...n]` с равни дължини. Числата `X[k]` и `Y[k]` определят местоположението на `k`-тия нефтен кладенец: абсцисата `X[k]` определя местоположението на кладенеца в посока изток-запад, а ординатата `Y[k]` определя местоположението на кладенеца в посока север-юг.

Алгоритъмът трябва да връща едно число — ординатата (`Y`) на нефтопровода.

Опишете идеята на алгоритъма словесно и дайте обосновка за нейната правилност. (Обосновката е математическа — чрез формули. Няма нужда от инварианти и др. под.)

Решението се зачита само ако предложеният алгоритъм е верен, достатъчно бърз и с валидна обосновка!

Задача 3. Имаме n предмета, чиито тегла са две по две различни. Някои от тях са претеглени по двойки (на везни с две рамена). Резултатите от претеглянето са записани като наредени двойки: (i, j) означава, че предмет № i е по-лек от предмет № j.

а) Да се провери дали информацията е противоречива.

б) Да се подредят предметите по тегло. Ако информацията не стига за еднозначното определяне на реда на предметите, да се изведе съобщение, съдържащо двойка (i, j) от номера на предмети, чиито тегла не могат да бъдат сравнени нито пряко, нито косвено.

Двете подусловия да се решат наведнъж — с един алгоритъм. Алгоритъмът трябва да има линейна времева сложност в най-лошия случай.

Задача 4. За една булева формула казваме, че е в *дизюнктивна нормална форма*, когато тя е дизюнкция от конюнкции, а операндите на конюнкциите са само отделни променливи или техните отрицания, като всяка променлива участва в една конюнкция най-много веднъж — или със, или без отрицание. Операндите на дизюнкцията (т.е. конюнкциите) се наричат клаузи на дизюнктивната нормална форма. Например

$$(\overline{a} \wedge b \wedge \overline{c}) \vee (b \wedge c) \vee (a \wedge \overline{b} \wedge \overline{c})$$

е дизюнктивна нормална форма с три променливи и три клаузи.

В горното определение се допуска формулата да бъде празна — без нито един символ. Приемаме, че такава формула също е в дизюнктивна нормална форма, тъй като съответства на празна дизюнкция (с нула клаузи).

Разглеждаме алгоритмичната задача за разпознаване DNF-SAT:

— Вход: булева формула F в дизюнктивна нормална форма.
— Въпрос: Удовлетворима ли е булевата формула F, тоест съществува ли поне един набор от стойности на променливите, при който F приема стойност "истина"?

а) Намерете класа на времева сложност на задачата DNF-SAT: **P**, **NP-с** и т.н. Отговорът да се обоснове подробно!

б) По-долу е изложена редукция на задачата SAT до задачата DNF-SAT.

> Разглеждаме алгоритмичната задача SAT за разпознаване дали е удовлетворима дадена конюнктивна нормална форма G. Задачата SAT се свежда до DNF-SAT по следния начин: с помощта на дистрибутивния закон на логиката превръщаме дадената формула от конюнктивната нормална форма G в еквивалентна на нея формула в дизюнктивна нормална форма F. Например
>
> $$(a \vee b \vee \overline{c}) \wedge (\overline{a} \vee \overline{b}) \iff (a \wedge \overline{b}) \vee (b \wedge \overline{a}) \vee (\overline{c} \wedge \overline{a}) \vee (\overline{c} \wedge \overline{b})\ .$$
>
> Тъй като двете формули са еквивалентни, то те приемат едни и същи стойности за едни и същи стойности на променливите. В частност, ако едната формула приема стойност "истина", то и другата формула приема стойност "истина". Иначе казано, G е удовлетворима, ако и само ако F е удовлетворима. Успяхме да сведем задачата SAT до задачата DNF-SAT, тоест SAT $\propto$ DNF-SAT.

Как се отнасяте към горните разсъждения? Изберете **<u>само един</u>** от следните отговори:

— Цитираните разсъждения са валидно доказателство, че SAT $\propto$ DNF-SAT. Тъй като задачата SAT е **NP**-трудна, то следва, че и задачата DNF-SAT също е **NP**-трудна.
— Цитираните разсъждения са валидно доказателство, че SAT $\propto$ DNF-SAT. Тъй като в подусловие "а" съм доказал(а), че задачата DNF-SAT $\in$ **P**, то следва, че и задачата SAT $\in$ **P**. Понеже задачата SAT е **NP**-пълна, то следва, че **P** = **NP**.
— Цитираните разсъждения <u>не</u> са валидно доказателство, че SAT $\propto$ DNF-SAT.

Ако приемате първия отговор за верен, дайте интуитивно обяснение по каква причина задачата DNF-SAT не може да се реши за полиномиално време.

Ако приемате втория отговор за верен, обяснете накратко как всяка задача от **NP** може да се реши за полиномиално време с помощта на задачите SAT и DNF-SAT.

Ако приемате третия отговор за верен, посочете пропуск в цитираните разсъждения.

РЕШЕНИЯ

Задача 1. Алгоритъмът STUPIDSORT се извиква рекурсивно три пъти върху подмасиви с дължина $2/3$ от първоначалната. Следователно времевата сложност $T(n)$ на алгоритъма удовлетворява рекурентното уравнение

$$T(n) = 3T\left(\frac{2n}{3}\right) + \Theta(1),$$

където събираемото $\Theta(1)$ съответства на нерекурсивната част от алгоритъма — сравнението на стойността на n с числото 2 и пресмятането на стойността на m.

Рекурентното уравнение може да се запише във вида

$$T(n) = 3T\left(\frac{n}{3/2}\right) + \Theta(1),$$

откъдето решението се получава с помощта на мастър-теоремата: $T(n) = \Theta\left(n^{\log_{3/2} 3}\right)$.

Това е времевата сложност на алгоритъма STUPIDSORT.

Сортирането чрез вмъкване изразходва време $\Theta\left(n^2\right)$ в най-лошия случай. Сравняваме двете сложности с помощта на граничен преход:

$$\lim_{n\to\infty} \frac{n^{\log_{3/2} 3}}{n^2} = \lim_{n\to\infty} n^{-2+\log_{3/2} 3} = +\infty,$$

защото $\log_{3/2} 3 > 2$, което следва от неравенството $\left(\frac{3}{2}\right)^2 = \frac{9}{4} < \frac{12}{4} = 3.$

Щом границата на частното е $+\infty$, то числителят расте по-бързо от знаменателя, тоест $n^{\log_{3/2} 3} \succ n^2$. Значи, сортирането чрез вмъкване е на порядък по-бързо от STUPIDSORT.

Задача 2. Нека има N кладенеца на север и M кладенеца на юг от нефтопровода. Отначало нефтопроводът се намира някъде на юг, тъй че всички нефтени кладенци са на север от него, тоест $N = n$, $M = 0$. Постепенно придвижваме нефтопровода на север. Ако го преместим на разстояние d на север, то общата дължина на тръбите, които свързват нефтопровода с кладенците на север от него, ще намалее с Nd, а пък общата дължина на тръбите, които свързват нефтопровода с кладенците на юг от него, ще се увеличи с Md. Тъй като $N > M$, сборът от дължините на всички свързващи тръби ще намалее с $(N - M)d$.

Като придвижваме нефтопровода на север, все повече кладенци се оказват на юг от него, тоест N намалява, а M расте. Докато $N > M$, изводът остава в сила: сборът от дължините на всички тръби намалява, когато придвижваме нефтопровода на север.

Когато M стане по-голямо от N, изводът се променя така: по-нататъшното придвижване на нефтопровода на разстояние d на север не само че не намалява, а напротив — увеличава общата дължина на свързващите тръби: тя нараства с $(M - N)d$.

Следователно общата дължина на тръбите ще е най-малка тогава, когато $M = N$. Тоест трябва да има равен брой кладенци на север и на юг от нефтопровода. Затова за ординатата Y на нефтопровода взимаме медианата на ординатите Y[1...n] на кладенците. Можем да намерим медианата за линейно време $\Theta(n)$ с помощта на алгоритъма PICK.

Условието на тази задача е преведено от книгата “Introduction to Algorithms” с автори Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Thomas H. Cormen.

Задача 3. Представяме информацията за извършените претегляния чрез ориентиран граф: върховете на графа съответстват на предметите (n на брой), а ребрата — на извършените претегляния (m на брой). Ребрата са ориентирани от по-лекия към по-тежкия предмет.

а) Информацията за претеглянията е противоречива само тогава, когато е невъзможно да се подредят предметите. Това се случва само когато се получи верига от неравенства, започваща и завършваща с един и същи предмет: $a < b < c < \ldots < d < a$, защото така излиза, че един предмет (в примера това е предметът a) е по-лек от себе си, което е невъзможно. Такава верига съответства на ***ориентиран цикъл в графа.*** Такъв цикъл се открива най-бързо, за линейно време $\Theta(m+n)$, с помощта на алгоритмичната схема ***търсене в дълбочина.*** Съответният алгоритъм е изучен на лекциите.

б) Подреждането на предметите по тегло може да се извърши за линейно време $\Theta(m+n)$ чрез известния алгоритъм за ***топологично сортиране.*** Той също се основава на схемата ***търсене в дълбочина,*** затова може да се изпълни едновременно с алгоритъма от точка "а". На практика това е един алгоритъм, който по време на подреждане на върховете на графа проверява за наличието на цикъл. Ако намери цикъл, алгоритъмът спира работа, тъй като предметите не могат да бъдат подредени по тегло: входните данни съдържат противоречие.

Ако входните данни са непротиворечиви, то графът не съдържа ориентиран цикъл и алгоритъмът намира някаква наредба на върховете, съвместима с входните данни. Тя обаче може да не е единствена. Ако има друга наредба, то входните данни са недостатъчни за еднозначното определяне на реда на предметите. Проверката за еднозначност става така: алгоритъмът обхожда върховете на графа в получения сортиран ред

$$v_1 \longrightarrow v_2 \longrightarrow v_3 \longrightarrow \ldots \longrightarrow v_n$$

и за всеки два последователни върха v_i и v_{i+1} проверява дали графът съдържа ребро от v_i към v_{i+1}. Тази проверка изисква едно обхождане на списъка $\mathrm{Adj}(v_i)$ на ребрата, излизащи от върха v_i, а общо за целия граф — обхождане на всички ребра. Понеже обхождаме и всеки връх по веднъж, то общото време е линейно: $\Theta(m+n)$.

Ако всички проверки минат успешно, то има $n - 1$ претегляния, задаващи следните съотношения на теглата: $v_1 < v_2 < \ldots < v_n$. Редът на предметите е еднозначно определен.

Ако някоя от проверките излезе неуспешна, тоест за някое i липсва ребро от v_i към v_{i+1}, то можем да разменим v_i и v_{i+1}. Новата наредба е съвместима с входните данни, тоест редът на предметите не е еднозначно определен. Алгоритъмът извежда двойката $\{v_i, v_{i+1}\}$.

Задача 4. Алгоритмичната задача за разпознаване $\mathtt{DNF\text{-}SAT} \in \mathbf{P}$, защото се решава за полиномиално време. По-точно, за константно време $\Theta(1)$: ако дадената булева формула (в дизюнктивна нормална форма) е празна, то алгоритъмът връща *false*, иначе връща *true*.

Обосновка за коректност: Празната дизюнкция е тривиално невярна, откъдето следва, че е неудовлетворима, затова алгоритъмът правилно връща *false*, ако входът е празна формула. Обратно, алгоритъмът правилно връща *true*, ако му бъде подадена непразна булева формула (в дизюнктивна нормална форма), защото всяка такава формула е удовлетворима: тя получава стойност "истина", ако изберем произволна клауза и зададем стойност "истина" на онези нейни променливи, които нямат отрицание; "неистина" — на променливите от същата клауза, които имат отрицание; произволни стойности — на останалите променливи.

Разсъжденията от условието <u>не</u> доказват, че $\mathtt{SAT} \propto \mathtt{DNF\text{-}SAT}$, тоест верен е третият от предложените отговори. Знакът $\propto$ означава полиномиална редукция, а редукцията от условието не е полиномиална, защото разкриването на скобите и новото комплектуване на променливите — по една от всяка клауза на дадената конюнктивна нормална форма — може да е много бавно. Ако тя съдържа например k клаузи с по две променливи, то след разкриването на скобите се получават 2^k конюнкции, т.е. изисква се експоненциално време.