ДОМАШНО № 3 ПО "ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ"
ЗА СПЕЦИАЛНОСТ "КОМПЮТЪРНИ НАУКИ", 2. КУРС, 1. ПОТОК
(СУ, ФМИ, ЛЕТЕН СЕМЕСТЪР НА 2018 / 2019 УЧ. Г.)

За предложените алгоритмични задачи установете дали са полиномиални, или са NP-пълни. Ако смятате, че някоя задача е полиномиална, съставете алгоритъм с полиномиална времева сложност, опишете го с думи или на псевдокод и анализирайте неговата времева сложност. Ако смятате задачата за NP-пълна, съставете полиномиална редукция и полиномиален алгоритъм за проверка на предложено решение, опишете ги (псевдокодът е задължителен за редукцията), анализирайте сложностите им по време и докажете коректността на редукцията.

Задачи 1 и 2 — играта "Digger"

В известната игра "Digger" един миньор се движи по игралното табло и събира смарагди. Всеки смарагд носи на миньора определен брой точки (един и същ за всички скъпоценни камъни). При взимане на смарагд прозвучава нота от октавата: долно до, ре, ми, фа, сол, ла, си, горно до. При първо взимане на смарагд се изпълнява долно до. Следваща нота (ре, ми, фа и т.н.) прозвучава само ако на предишния ход миньорът е взел смарагд. Когато миньорът премине през празно поле, октавата се прекъсва. От следващия смарагд започва нова октава (т.е. изпълнява се долно до).

Колкото повече смарагди миньорът вземе последователно, толкова по-дълга част от октавата звучи. Ако миньорът вземе смарагди с осем последователни хода, т.е. ако изпълни цялата октава, получава бонус (допълнителни точки). При взимане на следващ смарагд започва нова октава — изпълнява се пак долно до, независимо дали е взет веднага след осемте.

Миньорът иска да получи възможно най-много точки от всяко ниво на играта. Затова трябва да организира ходовете си така, че октавата да прозвучи докрай колкото може повече пъти.

Моделираме всяко ниво чрез ориентиран граф $G(V, E)$ с $|V| = n$ върха и $|E| = m$ ребра:

— Върховете на графа са полетата на игралното табло.

— Ребрата на графа съответстват на възможните ходове по таблото.

— Всеки връх притежава маркер — дали на съответното поле има смарагд.

— Посочен е един връх s — полето, на което се намира миньорът в началото на играта.

Миньорът тръгва от върха s и се движи по ребрата на графа. През един и същ връх на графа миньорът може да премине колкото пъти иска, но ако във върха има смарагд, миньорът го взима задължително при първото преминаване, след което върхът остава празен (без смарагд).

Разглеждаме две алгоритмични задачи за разпознаване с еднакъв вход: $G(V, E)$, n, m, s. Задачите се различават само по въпроса: Възможно ли е миньорът да се движи така, че:

— поне веднъж да се изпълни цялата октава (задача 1) ?

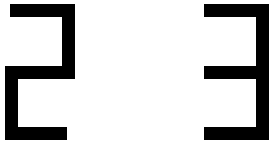
— да вземе всички смарагди, без да прекъсне никоя октава (задача 2) ?

В задача 2 се пита дали е възможно миньорът да вземе всички смарагди на групи по осем. За целта трябва броят k на смарагдите да се дели на 8 и октавата да бъде изцяло изпълнена $k / 8$ пъти.

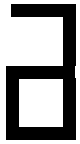
Намерете класовете на времева сложност на задача 1 и задача 2.

Задача 3 — опаковане на графи

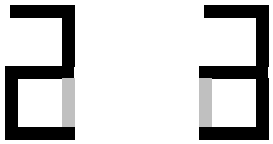
Знаците от някакъв шрифт се изписват само с прави черти, тоест всеки знак се изобразява като начупена линия. Искаме да създадем светлинно табло, което да може да показва всички знаци от този шрифт. Таблото трябва да бъде възможно най-евтино, тоест да има възможно най-малко светещи отсечки. За простота да разгледаме шрифт, съставен само от два знака, например 2 и 3.



Всеки от двата знака се състои от пет отсечки. Следователно табло с общо 5 + 5 = 10 отсечки ще свърши работа. Само че този брой (десет) е излишно голям. Шест отсечки са достатъчни, както се вижда от показания тук пример.



На това табло можем да изобразим всяка от цифрите 2 и 3, като включим едни отсечки да светят, а другите оставим изключени. Черните черти на чертежа означават светещи отсечки на таблото, а сивите черти — изключени, тоест несветещи, отсечки.



В такава ситуация е естествено да използваме теорията на графите: върховете на начупената линия ще бъдат върхове на граф, а отсечките на начупената линия — ребра на графа.

Друга ситуация, различна на пръв поглед, която се свежда до графи по аналогичен начин: Искаме да образуваме работен екип измежду множество наши познати. Някои от тях се познават, а други — не. Екипът трябва да се състои от няколко подгрупи, като се допуска един и същи човек да бъде член на няколко подгрупи. Отново да приемем за простота, че подгрупите са само две. Едната подгрупа да се състои от началник и трима подчинени: началникът е длъжен да познава подчинените си (за да знае на кого какви задачи да възлага), но те не бива да се познават един друг (за да не правят заговори срещу шефа). Другата работна група трябва да бъде шпионска верига от четирима души: първият ще събира сведения и ще ги предава на втория, вторият — на третия, третият — на четвъртия, четвъртият — на нас. С цел секретност (ако някой от тях бъде разкрит, да не се стига лесно до останалите) искаме първият да познава само втория член на веригата, вторият — само първия и третия, третият — само втория и четвъртия, четвъртият — само третия. Пита се колко души най-малко трябва да вземем в екипа.

Тази ситуация може да бъде изобразена така:

Подгрупата, съставена от началник и трима подчинени, прилича на буквата Т: точката, от която се разклоняват трите отсечки, е началникът, а трите върхчета на буквата са неговите подчинени. Шпионската верига може да се изобрази като буквата П: четиримата агенти съответстват на четирите върхчета на буквата.

Можем да съставим екип от 4 + 4 = 8 души, но това е излишно много. Петима членове са достатъчни, ако техните познанства образуват полегналата буква F.

Тази конфигурация съдържа двете желани подгрупи — буквите Т и П.

Примерите по-горе са описания на практически ситуации, които могат да се формализират по различни начини. За да моделираме задачата, ще използваме средства от теорията на графите. Една възможна формулировка гласи:

По дадени неориентирани графи G_1, G_2, ..., G_k да се построи възможно най-малък неориентиран граф G, който съдържа като подграфи всички дадени графи G_1, G_2, ..., G_k.

Тази формулировка добре отразява свързването на обектите (топологичната информация). Това стига за втората ситуация (екипа с подгрупите), но не и за първата (знаците от един шрифт): губим ориентацията на знаците. Например 8 и ∞ са различни знаци, но им съответства един граф. Аналогично, П и С са различни букви, но ако ги изобразим с отсечки, ще изглеждат еднакво, само че завъртени под прав ъгъл една спрямо друга.

Въпреки посочения недостатък все пак ще използваме графи, за да не усложняваме задачата с допълнителни сведения за ориентацията на обектите в пространството, още повече че понякога липсва такава информация (например в задачата за екипа и подгрупите).

Има и друг пропуск: не е казано в какъв смисъл графът G е най-малък. Това може да бъде уточнено по различни начини. Например в първата описана ситуация (знаците от един шрифт) графът G трябва да съдържа възможно най-малко ребра (чертичките на светлинното табло), а пък във втората ситуация (екипът и подгрупите) G трябва да има възможно най-малко върхове (тоест екипът да съдържа възможно най-малко членове).

Можем да опитаме минимизация по двата критерия едновременно — по броя на върховете и по броя на ребрата на графа. Сега задачата гласи:

По дадени неориентирани графи G_1, G_2, ..., G_k да се построи неориентиран граф G с най-малко върхове и ребра, който съдържа като подграфи всички графи G_1, G_2, ..., G_k.

Дадените примери показват, че с тази задача можем да моделираме разнообразни ситуации от практиката. Възниква въпросът има ли бърз (тоест полиномиален) алгоритъм. За съжаление, се оказва, че задачата е NP-пълна, затова не съществува полиномиален алгоритъм (ако P ≠ NP). Нещо повече, задачата остава NP-пълна дори в частния случай $k = 2$. Ето защо ще разгледаме само този частен случай. (Това е достатъчно: щом частният случай е NP-труден, то следва, че и общият случай е такъв.)

Твърдението за NP-пълнота е неточно: само задачите за разпознаване могат да са NP-пълни, а нашата задача е оптимизационна. Когато казваме, че тя е NP-пълна, подразбираме, че NP-пълна е всъщност съответната задача за разпознаване. Нека я формулираме изрично. Това ще бъде окончателната и най-точна формулировка (разглеждаме само частния случай $k = 2$).

Определяме следната задача за разпознаване, да я наречем *Опаковане на графи*:
— Вход: Дадени са неориентираните графи G_1 и G_2 (това означава, че са дадени също броят на техните върхове n_1 и n_2 и броят на техните ребра m_1 и m_2 — четири цели неотрицателни числа). Дадени са още две цели неотрицателни числа n и m.
— Въпрос: Съществува ли неориентиран граф G с не повече от n върха и не повече от m ребра, който съдържа подграф, изоморфен на G_1, и подграф, изоморфен на G_2 ?

Докажете, че алгоритмичната задача *Опаковане на графи* е NP-пълна.