Второй курс, осенний семестр 2020/21
Практика по алгоритмам #5

Mincost Flow
6 октября

Собрано 4 октября 2021 г. в 18:20

Содержание

Mincost Flow

1. **Mincost flow**

   Найдите mincost поток в орграфе. Размер потока не важен.

   a) Сведите задачу к mincost circulation.
   b) Сведите задачу к mincost k-flow. Бинпоиском и без бинпоиска.
   c) Подумайте, отличается ли $c_e, w_e \in \mathbb{Z}$ от $c_e, w_e \in \mathbb{R}$.

2. **LR через mincost**

   Найти LR-поток, используя mincost flow, за время $\mathcal{O}(\texttt{mincost flow})$.

3. **Транспортная задача**

   В городе есть дороги, заводы-производители и магазины-дистрибьюторы. Дороги образуют орграф, у каждой дороги есть длина w_i и пропускная способность u_i. Каждый день i-й завод выпускает A_i единиц товара, j-й магазин продаёт B_j единиц товара. Составьте план доставки товара от заводов к магазинам так, чтобы $\sum f_i w_i$ пройденных дорог была минимальна.

   Рассмотрим жизненную версию: $u_i = +\infty$ (по дороге можно пустить сколько угодно грузовиков, стоимость потока = стоимость бензина на перевозку). Как проще можно решить такую задачу? Как думаете, есть ли жадное решение?

4. **Mincost + Диниц**

   Помните, как Эдмондс-Карп улучшился до Диница? Помните, что mincost k-flow – почти Эдмондс-Карп, но Форд-Беллман вместо bfs? Попробуем улучшить алгоритм для mincost.

   Рассмотрим такой алгоритм для поиска mincost потока: пока $\exists$ дополняющий путь в G_f, выделяем сеть кратчайших путей в G_f, ищем максимальный поток в этой сети.

   a) Докажите корректность алгоритма.
   b) За сколько работает такой алгоритм?
   c) К каким графам его разумно применять?

5. **Подпоследовательности**

   Дан массив целых чисел, выбрать в нём k возрастающих непересекающихся подпоследовательностей. максимальной суммарной длины.

   a) $k = 1$. Простое решение за $\mathcal{O}(n \log n)$.
   b) $k = 2$. Простое решение.
   c) $k = 2$. Решение потоками.
   d) $k = 2$. Решение потоками за $\mathcal{O}(n^2)$.
   e) $\forall k$ решение за $\mathcal{O}(kn^2)$.
   f) Обобщение: дан ацикличный орграф, у каждой вершины есть вес, выбрать k вершинно непересекающихся простых путей так, чтобы сумма весов выбранных вершин была максимальна. $\mathcal{O}(E + kV^2)$.
   g) Что изменится для графа с циклами? Подумайте про $k = 1$. Что не так в рассуждении «mincost же работает, наш ответ корректен»?

6. **Задача про автоматы**

   Есть k автоматов и n заданий. Про каждое задание известно, во сколько его нужно начать делать, во сколько закончить, а также его стоимость. Каждый автомат может выполнять только одно задание в каждый момент времени. Нужно выполнить задания максимальной суммарной стоимости.

   a) Решение за $\mathcal{O}(kn^2)$.
   b) Решение за $\mathcal{O}(kn \log n)$.
   c) (*) Уменьшите число вершин в два раза.

7. **Вершинно-взвешенное паросочетание**

   Дан двудольный граф. У вершин есть неотрицательные веса. Вес ребра равен сумме весов его концов. Найти паросочетание максимального веса.

   a) $\mathcal{O}(V^3)$.
   b) $\mathcal{O}(VE)$. dfs от алгоритма Куна.

8. **Min mean cycle canceling**

   $c_e \in \mathbb{R}^+, w_e \in \mathbb{R}$.
   Если в остаточной сети нет отрицательных рёбер, mincost circulation можно найти за $\mathcal{O}(1)$.
   Мы можем менять веса рёбер в остаточной сети двумя способами:
   применить потенциалы, пустить по циклу поток.

   Обозначим w_e^p – вес ребра с учётом потенциала p, G^p – граф с пересчитанными весами.

   a) Какие потенциалы p нужно применить, чтобы $\min_e w_e^p \to \max$ (обозначим $\max = x$)?
   b) Найдём min mean cycle, толкнём по нему поток. Что произойдёт с x?
   c) Посмотрите на G_p: $w_e^p \geqslant 0$. Через сколько итераций x должен увеличиться?
   d) Посмотрите на G с исходными весами. Через сколько итераций min mean cycle пройдёт по $w_e \geqslant 0$? Докажите, что через m итераций x увеличится хотя бы в $1 - \frac{1}{n}$ раз.
   e) Оцените время поиска mincost циркуляции с помощью MMCC.

9. (*) **Непрерывная цена, мультипаросочетание**

   Дан двудольный граф. Нужно найти мультипаросочетание – выбрать подмножество рёбер, у каждой вершины ограничена степень. Запишем задачу формально.
   $f_{ij} \in \mathbb{Z}$ – поток между i-й вершиной первой доли и j-й вершиной второй доли.
   $0 \leqslant f_{ij} \leqslant 1$, $s_i = \sum_j f_{ij} \leqslant a_i$, $t_i = \sum_j f_{ij} \leqslant b_j$.

   Пусть стоимость мультипаросочетания равна $\sum_i cost_i \cdot s_i^2$ (в обычном mincost потоке $\sum_i cost_i \cdot s_i$). Минимизируйте стоимость максимального по $\sum_i s_i$ (размеру) паросочетания.

10. (*) **Регионы памяти. Расписание выполнения программ.**

    a) Есть k регионов памяти и n программ. У каждого региона есть размер s_i. У каждой программы есть необходимое ей количество памяти x_j и время выполнения t_j. Каждой программе нужно сопоставить номер региона памяти i_j, в котором она будет выполняться, и отрезок времени выполнения $[l_j, l_j{+}t_j{=}f_j)$. Для каждого региона памяти отрезки времени выполнения программ не должны пересекаться. Минимизировать $\sum_j f_j$.
    b) Усложним задачу. Теперь у каждой программы есть вектор пар $\langle x_j, t_j \rangle$ –
    если предоставить программе хотя бы x_j памяти, она будет выполняться t_j времени.

11. **(*) Кредитные операции - 2 (повторим)**

    По заданной матрице a_{ij} найти такие вектора x и y, что $x_i + y_j \geqslant a_{ij}$, а $\sum x_i + \sum y_j \to \min$. Дополнительно известно, что матрица или квадратная, или неотрицательная.

12. **(**) Котики должны спать и есть.**

    Дано $n \leqslant 1000$ минут. Каждую минуту кот или ест, или спит (всю минуту).
    В каждые k подряд идущих минут кот должен есть хотя бы e минут, спать хотя бы s минут.

    Даны числа $x_i, y_i \geqslant 0$. Постройте расписание для кота, максимизирующее $\sum_{i \in S} x_i + \sum_{i \in E}$, где S – множество минут, когда кот спит, а E – множество минут, когда кот ест.

Разбор задач практики

1. **Mincost flow**

   Сведение к mincost circulation: добавить ребро $e\colon t \to s,\ c_e = +\infty,\ w_e = 0$.

   Сведение к mincost k-flow. Пусть f_k – оптимальный поток размера k, $path_k$ — кратчайший путь в остаточной сети G_{f_k}. Вспомним, что $W(path_k)\nearrow \Rightarrow$ найдём $\min k\colon W(path_k) \geqslant 0$.

   Можно искать k бинпоиском, можно линейным (толкая $\min_e(c_e - f_e)$ по очередному пути). Линейный поиск даст время $\mathcal{O}(VE \cdot \texttt{Dijkstra})$, также как и простейший mincost k-flow. Внутри бинпоиска можно использовать более крутые решения. Например, спера Диниц с link-cut ищет любой поток h размера k, затем capacity scaling ищет mincost circulation в $G_h \Rightarrow \mathcal{O}(VE\log V + \texttt{Dijkstra} \cdot E\log U)$.

   Если $c_e \in \mathbb{R}$, то log бинпоиска и log от capacity scaling становятся вида $\log\frac{U}{\varepsilon}$.

2. **LR через mincost**

   Вместо LR-ребра делаем L-ребро стоимости -∞ и $(R-L)$-ребро стоимости 0. Если в полученном графе есть отрицательные циклы – ничего страшного, задачу mincost circulation мы тоже умеем решать.

3. **Транспортная задача**

   Как обычно, перенаправим избытки (A_i у заводов) в недостатки (B_j у магазинов). Для этого добавляем исток, сток и ищем mincost max flow.

   Если $\forall e\ u_e = +\infty$, можно найти матрицу расстояний, составить двудольный граф. Если $A_i = B_j = 1$, получаем задачу о назначениях $\Rightarrow$ mincost flow или венгерка, но не жадность.

4. **Mincost + Диниц**

   Пусть перед очередной фазой наш поток – f, G_f – остаточная сеть,
   $H(G_f)$ – сеть кратчайших путей. $d(s,t)$ – длина кратчайшего пути от s до t в G_f.

   Корректность.
   $\forall$ путь из s в t в $H(G_f)$ – кратчайший из s в t в G_f. Мы уже знаем, что $d(s,t)\nearrow$.
   Пусть Δf = max поток в $H(G_f)$, декомпозируем его на пути, все они веса d, циркуляция в декомпозиции неотрицательна $\Rightarrow W(f + \Delta f)$ оптимален.

   Время работы.
   В отличии от Диница сеть кратчайших путей $H(G_f)$ – не слоистый граф, а произвольный. Например, пусть $\forall e\ w_e = 0 \Rightarrow H = G_f$ (сеть кратчайших путей содержит все рёбра G_f) $\Rightarrow$ нельзя пустить одну фазу Диница, нужно пускать полноценный поток.
   После одной фазы «пустим Диница» длина кратчайшего пути возрастёт (от противного): $G_{f+\Delta f}$ содержит путь p длины $d \Rightarrow$ рассмотрим поток $(\Delta f) + p$ в G_f, декомпозируем его на пути, все пути имеют длину ровно $d \Rightarrow (\Delta f) + p$ – поток в $H(G_d) \Rightarrow \Delta f$ не максимален.
   $\Rightarrow$ время работы $\mathcal{O}(D(VE + \texttt{maxFlow}))$, где D – число различных кратчайших расстояний.

   Работает быстро, если D мало.
   Если все цены рёбер в графе равны, различных расстояний не более V, получается $\mathcal{O}(V(VE + \texttt{maxFlow}))$. Если все цены маленькие, $\mathcal{O}(1)$, выйдет такая же асимптотика.

5. **Подпоследовательности**

   Решение динамикой для $k = 1$. `minEnd[length]` за $\mathcal{O}(n \log n)$.

   Решения динамикой для $k = 2$.
   `sumLength[end1,end2]` за $\mathcal{O}(n^3)$, `minEnd2[end1,sumLength]` за $\mathcal{O}(n^2 \log n)$.

   Mincost flow.
   Раздваиваем вершины, $e\colon i \to j, i < j, a_i < a_j, w_e = -1, c_e = 1$. Время работы $k \cdot VE = k \cdot n^3$.

   Дейкстра. Исходный граф ацикличен $\Rightarrow$ можно за $\mathcal{O}(V + E)$ динамикой (dfs) найти кратчайшие пути, использовать их, как потенциалы, далее пускать Дейкстру. $\mathcal{O}(kn^2)$.

   Ацикличный граф. Раздваиваем вершины, помним про dfs для поиска кратчайших путей, пускаем поток Дейкстрой.

   Что не так с циклами? Для $k = 1$ нас просят найти гамильтонов путь.

6. **Задача про автоматы**

   Можно решать так же, как предыдущую: ребро между заданиями, если одно можно начать после конца другого. Получится граф из $\Theta(n^2)$ ребер.

   Быстрое решение. Вершины графа – моменты времени: начала и концы заданий. Ребро цены $-cost_i$ и пропускной способности 1 между началом и концом i-го задания. Также ребро из каждого момента времени в следующий момент, цена 0, пропускная способность $+\infty$. То есть, свободный автомат либо берет задание и освобождается к концу задания, либо переходит в следующий момент времени, оставаясь свободным. Получилось $2n$ вершин и $3n$ рёбер. Mincost поток размера k ищем за $\mathcal{O}(E + k \cdot Dijkstra) = \mathcal{O}(kE \log V) = \mathcal{O}(kn \log n)$.

   Ещё меньше вершин. Заметим, что из второй половинки задания всегда исходит ровно одно ребро. Зачем нам такая вершина? Получаем n вершин и $2n$ рёбер.

7. **Вершинно-взвешенное паросочетание**

   Венгерка и mincost дают $\mathcal{O}(V^3)$. Внутри mincost важно использовать Дейкстру.

   Чтобы получить решение за $\mathcal{O}(VE)$ заметим, что вес дополняющего пути равен сумме весов конечных вершин. Давайте за $\mathcal{O}(E)$ найдём пару вершин $i, j\colon a_i + b_j = \min$ и есть путь $i \rightsquigarrow j$. Перебираем i в порядке a_i и для каждого i ищем из неё путь в dfs-ом в минимальное b_j, игнорируя те j, которые мы нашли для предыдущих i (т.е. не обнуляя пометки между dfs).

8. **Min mean cycle canceling**

   `[Lecture in english]`

   $x =$ весу цикла минимального среднего веса.

   Как найти потенциалы? Найти x, и в графе $w'_e = w_e - x$ (без отрицательных циклов) найти расстояния, которые и есть потенциалы.

   x может только увеличиться.

   В сети G_p с весами $w'_e = w_e - x$ нет отрицательных рёбер и каждый цикл будет насыщать хотя бы одно нулевое ребро $\Rightarrow$ за m итераций нулевые рёбра кончатся.

   В сети G_p с весами w_e нет рёбер меньше $x \Rightarrow$, текущий цикл минимального среднего веса имеет все рёбра равные $x \Rightarrow$ когда очередной цикл пройдёт через ребро веса $w_e > 0$, вес цикла будет хотя бы в $1 - \frac{1}{length}$ раз больше.

9. (*) **Непрерывная цена, мультипаросочетание**

Когда мы уже пустили f потока в i-ю вершину левой доли, мы платим $f^2 cost_i$. Пустим ещё +ε потока, получим стоимость $(f+\varepsilon)^2 cost_i$, прирост равен $(2f\varepsilon+\varepsilon^2)cost_i = 2f\varepsilon cost_i + o(\ldots)$. Получаем, что удельная стоимость единицы потока равна $2f \cdot cost_i$.

Можно представить, что у нас не одно ребро, а бесконечно много узких рёбер. j-е ребра ребра в i-ю вершину левой доли стоит $2(\varepsilon \cdot j) \cdot cost_i$. В одной пачке рёбра упорядочены по возрастанию веса ($cost_i > 0$). Стоимость кратчайшего пути – стоимость его первого ребра. Алгоритм поиска mincost потока говорит, что корректно +ε потока толкнуть по пути, который начинается с ребра минимального веса.

Решение: n раз бинпоиском находим такое максимальное x, что можно пустить по всем рёбрам поток $\frac{x}{2cost_i}$, таким образом, заплатив за каждое $\leqslant x$, $cost_i$ пересчитано в соответствии с уже пущенным потоком. Проверка внутри бинпоиска через max flow. После конца бинпоиска по одному из n рёбер в левую долю нельзя пустить даже $x+\varepsilon$ потока, его можно просто удалить. Всего из истока исходит n рёбер, в худшем случае они будут умирать по одному.

`Time` $= \mathcal{O}(n \log MAX \cdot MaxFlow)$.

10. (*) **Регионы памяти. Расписание выполнения программ.**

    ?

11. (*) **Кредитные операции - 2 (повторим)**

    ?

12. (**) **Котики должны спать и есть.**

    ?

Домашнее задание

3.1. Обязательная часть

1. **(2) Mincost поток нельзя искать bfs-ом**

   Рассмотрим задачу: найти в невзвешенном орграфе два рёберно непересекающихся пути из s в t минимальной суммарной длины. Мальчик Вася пытается решить задачу так «запустим два раза bfs». Покажите, что и если Вася подумал об обратных рёбрах, и если не подумал, его решение некорректно. По баллу за каждый из двух вариантов.

2. **(2) Оптимальность mincost потока**

   Рассмотрим задачу mincost k-flow. $n, m \leqslant 10\,000$. Предложите строго полиномиальный алгоритм, который проверяет оптимальность найденного ответа.

3. **(2.5) Mincost LR-flow**

   Предложите алгоритм для поиска LR-потока минимальной стоимости.

4. **(4) Равномерное паросочетание 2**

   Дана матрица выполнимости – какой рабочий какие работы может выполнять. Распределим **все** работы между рабочими. Пусть i-му рабочему досталось v_i работ. Рассмотрим вектор $V = (v_1, v_2, \ldots, v_m)$. Нужно так распределить работы, чтобы $|V_{opt} - V| \to \min$. $V_{opt} = (\frac{n}{m}, \frac{n}{m}, \ldots, \frac{n}{m})$, вектор вещественных чисел, где n – число работ, m – число рабочих, $|X - Y|$ – евклидово расстояние между m-мерными векторами X, Y.

   a) **(3)** Свести задачу к потоку минимальной стоимости.
   b) **(1)** Свести задачу к алгоритму Куна или Куноподобному dfs-у.

5. **(3) Автомойка**

   Есть n машин. Каждую нужно привести в приличный вид, для этого нужно обработать её в трёх мастерских в строго опеределённом порядке — сперва A, затем B, затем C.
   В день каждая мастерская может принять не более одной машины.
   Нужно уложиться в $m \geqslant n+2$ дней. A берёт в i-й день a_i за процедуру, $B \to b_i$, $C \to c_i$.
   Даны n, m, a_i, b_i, c_i, найти расписание, минимизирующее суммарную стоимость процедур.
   (3) $\mathcal{O}(Polynom(n, m))$
   (+1) $\mathcal{O}(nm \log m)$

3.2. Дополнительная часть

1. **(3) Подгон MST**

   Дан граф G, в нём выделено остовное дерево T. Мы можем уменьшать и увеличивать веса рёбер. Сделать T минимальным по весу остовным деревом. При этом минимизировать суммарное изменение весов рёбер.

   Подсказка: остов минимальный iff вес любого ребра не из остова не меньше максимума на соответствующем пути. Мы на практиках решали задачу про $x_i + y_j \geqslant a_{ij}$.

2. **(4) Увеличение длины пути**

   Дан орграф с неотрицательными весами рёбер.
   Стоимость увеличения веса ребра e на $x \in \mathbb{R}$ равна $c_e \cdot x$.
   Есть P денег, потратьте их с умом – максимизируйте стоимость кратчайшего пути из s в t.

3. **(3.5) Крестьяне и поля**

   Дана матрица $n \times m$. В некоторых клетках горы, в некоторых живут крестьяне, в некоторых поля. Расстояние между клетками (x_1, y_1) и (x_2, y_2) считается без учёта гор: просто $|x_1 - x_2| + |y_1 - y_2|$. Полей не меньше, чем крестьян. Если фиксирован порядок крестьян p, то можно раздать поля следующим алгоритмом: в порядке p каждый крестьянин получает свободное поле, из таких ближайшее, из таких $x \to \min$, из таких $y \to \min$.

   Задача: выбрать p, чтобы сумма расстояний от крестьян до их полей была минимальна.