Как записывать задачи?

Е. Поршнев

В этой лекции мы будем записывать решение такой задачи:

Задача 1. Сколько рёбер может иметь дерево? (Дерево — связный граф, не содержащий циклов.)

Перед тем, как записывать решение, неплохо бы его придумать. Тем не менее процесс придумывания решения выходит за рамки настоящей лекции, поэтому предположим, что как-то мы это сделали.

Итак, решение в том виде, в каком оно пришло в голову:

Идея решения.

Выберем какую-нибудь одну вершину дерева и будем идти из неё по рёбрам, пока получается. Когда мы зайдём в тупик, нужно будет перестроить исходное дерево, перенеся ребро, ведущее в одно из ответвлений, которое мы ещё не посетили, в конец нашего пути. В результате этого процесса получится ломаная, обходящая все вершины дерева и содержащая звеньев столько же, сколько рёбер в дереве. Значит, число рёбер дерева на 1 меньше числа его вершин.

Звучит весьма сумбурно и не понятно никому кроме автора идеи. Поэтому теперь попытаемся записать всё более развёрнуто.

Для начала введём какие-нибудь обозначения. Чтобы не думать долго, что понадобится в решении, а что нет, обозначим всё, что только можно. А потом лишнее уберём. Итак:

Обозначения.

G — исходное дерево
V — множество вершин G
E — множество рёбер G
$n = |V|$
$m = |E|$

Поскольку формулировка задачи содержит в себе вопрос, следующей частью нашего текста будет ответ. В первую очередь это нужно затем, чтобы читателю решения не приходилось искать его по всему тексту. Кроме того мысль гораздо легче проследить, когда с самого начала понятно, в каком направлении она движется.

Ответ.

$m = n - 1$.

Теперь пора переходить собственно к решению. Для начала на основе идеи решения составим общий план. Весь дальнейший текст будет рождаться именно как его детализация. Пока просматривается две логических части:

- *построение ломаной, обходящей дерево;*
- *вывод о том, сколько должно быть рёбер в G.*

Вынесем всё рассуждение про ломаную в отдельное утверждение:

Утверждение 1.

Дерево G можно так «перестроить», что получится ломаная, проходящая через все вершины G и содержащая в точности m рёбер.

Доказательство утверждения 1. (Попытка номер 1)

Выберем какую-нибудь одну вершину $r \in V$ и назовём её корнем дерева. Будем идти из r по рёбрам, пока получается. Когда мы зайдём в тупик, перестроим исходное дерево, перенеся ребро, ведущее в одно из ответвлений, которое мы ещё не посетили, в конец нашего пути. После такого переноса пройдём ещё сколько получится; затем при необходимости сделаем ещё один перенос и т. д. В результате описанного процесса получится ломаная, обходящая все вершины дерева и содержащая звеньев столько же, сколько рёбер в дереве.

Проанализируем недостатки получившегося текста. Сразу бросается в глаза, что процесс последовательной перестройки дерева описан очень схематично и (о ужас!) содержит слова «и так далее». Избежать этого можно двумя способами: оформив доказательство по индукции, либо чётко сформулировав алгоритм в виде, пригодном для компьютера. Мы пойдём вторым путём. Итак:

Доказательство утверждения 1. (Попытка номер 2)

Выберем какую-нибудь одну вершину $r \in V$ и назовём её корнем дерева. Будем строить ломаную, обходящую все вершины дерева G...

Хм... А ведь всё равно наша ломаная не всегда проходит по рёбрам G. Зачем же тогда вообще говорить о ломаной? Будем просто рассматривать последовательность вершин из G. Она будет начинаться с корня и мы будем иметь в виду, что соседние вершины нашей последовательности соединены звеньями ломаной. Попробуем-ка ещё раз.

Доказательство утверждения 1. (Попытка номер 3)

Выберем какую-нибудь одну вершину $r \in V$ и назовём её корнем дерева. Будем строить последовательность $(v_1, v_2, \ldots)$ вершин графа G по следующему алгоритму:

1. $v_1 = r$.
2. Если в какой-нибудь момент все вершины из V уже включены в последовательность, алгоритм завершается.
3. *(Удлинение)* Предположим, что на некотором шаге уже построена последовательность $(v_1, \ldots, v_k)$. Пусть найдётся вершина v, соединённая ребром с v_k и ещё не включённая в последовательность. Тогда возьмём $v_{k+1} = v$.
4. *(Перестройка)* Предположим, что на некотором шаге уже построена последовательность $(v_1, \ldots, v_k)$. Пусть теперь не найдётся вершины, соединённой ребром с v_k и ещё не включённой в последовательность. Выберем наибольшее число $i < k$ со следующим свойством: найдётся вершина v, соединённая ребром с v_i и ещё не включённая в последовательность. Существование такого i будет доказано ниже отдельно. Удалим из E ребро (v_i, v) и добавим вместо него ребро (v_k, v). Возьмём $v_{k+1} = v$.

Кстати: если внимательно посмотреть на наш алгоритм, становится видно, что априорной гарантии того, что дерево в какой-нибудь момент не перестанет быть деревом, у нас нет. А очень хотелось бы, чтобы после завершения алгоритма, у нас всё-таки осталось дерево. Придётся это тоже доказать. Заодно введём ещё несколько обозначений.

Доказательство утверждения 1. (Продолжение попытки номер 3)

Докажем, что в результате применения нашего алгоритма перестройки граф G не перестанет быть деревом. Обозначим через G_k граф, получившийся из G после применения k шагов алгоритма. Докажем по индукции следующее утверждение: G_k — дерево.

База:

В силу $G_1 = G$ база очевидна.

Шаг индукции:

Мы предполагаем, что G_k — дерево.

Если $(k+1)$-ый шаг — удлинение, то $G_{k+1} = G_k$. Значит, G_{k+1} — дерево.

Пусть $(k+1)$-ый шаг — перестройка. Рассмотрим какой-нибудь путь[1] P в графе G_k. Поставим ему в соответствие путь $\alpha(P)$ в графе G_{k+1} следующим образом:

- Если в P не встретилось ребро (v_{k+1}, v_i) — то самое, которое было удалено на $(k+1)$-ом шаге алгоритма, — берём $\alpha(P) = P$.
- Если в P встретилось ребро (v_{k+1}, v_i), заменим его на последовательность рёбер $(v_{k+1}, v_k), (v_k, v_{k-1}), \ldots, (v_{i+1}, v_i)$. Полученный путь возьмём в качестве $\alpha(P)$.
- Если в P встретилось ребро (v_i, v_{k+1}), заменим его на последовательность рёбер $(v_i, v_{i+1}), \ldots, (v_{k-1}, v_k), (v_k, v_{k+1})$. Полученный путь возьмём в качестве $\alpha(P)$.

И наоборот: рассмотрим какой-нибудь путь Q в графе G_{k+1}. Поставим ему в соответствие путь $\beta(Q)$ в графе G_k следующим образом:

- Если в Q не встретилось ребро (v_{k+1}, v_k) — то самое, которое было добавлено на $(k+1)$-ом шаге алгоритма, — берём $\beta(Q) = Q$.
- Если в Q встретилось ребро (v_{k+1}, v_k), то за ним должны следовать рёбра $(v_k, v_{k-1}), \ldots, (v_{i+1}, v_i)$ — это следует из выбора i и из того, что G_k — дерево. Заменим всю последовательность этих рёбер $(v_{k+1}, v_k), (v_k, v_{k-1}), \ldots, (v_{i+1}, v_i)$ на одно ребро (v_{k+1}, v_i). Полученный путь возьмём в качестве $\beta(Q)$.
- Аналогично, если в Q встретилось ребро (v_k, v_{k+1}), перед ним следуют рёбра $(v_i, v_{i+1}), \ldots, (v_{k-1}, v_k)$. Заменим их все на одно ребро (v_i, v_{k+1}). Полученный путь возьмём в качестве $\beta(Q)$.

Легко видеть, что отображения α и β взаимно обратны, поэтому каждое из них взаимно-однозначно. Кроме того, концы пути при отображениях α и β сохраняются. Значит, две вершины связаны единственным путём в графе G_{k+1} тогда и только тогда, когда они связаны единственным путём в графе G_k. Из этого следует, что G_{k+1} — дерево.

Докажем теперь, что число i, упоминаемое на шаге 4 алгоритма, действительно существует. Предположим, что это не так. Это значит, что в графе G_k ни одна из вершин, уже включённых в последовательность, не имеет соседей не из последовательности. А поскольку граф G_k связен (мы уже доказали, что это дерево), получаем, что все вершины G уже включены в последовательность. Значит, алгоритм должен был уже завершиться. Противоречие.

Докажем, что алгоритм закончит свою работу. Действительно, на каждом шаге алгоритма, число вершин в последовательности возрастает на 1 и при этом вершины в последовательности не повторяются. Поскольку число вершин в графе G конечно, алгоритм завершится.

Отметим также, что в результате производимых перестроек число рёбер в графе не изменяется. При этом, если алгоритм совершил в общей сложности k шагов, то граф G_k и будет искомой ломаной.

Утверждение 1 полностью доказано.

Во, так уже получше. Перед тем, как переходить ко второму пункту плана, подумаем, что ещё можно было бы исправить в утверждении 1? Во-первых, совершенно излишним оказалось понятие «корня» дерева. Мы его использовали только один раз при запуске алгоритма. Вместо этого можно написать так:

Доказательство утверждения 1. (Версия 3.1)

Будем строить последовательность $(v_1, v_2, \ldots)$ вершин дерева G по следующему алгоритму:

1. Вершину v_1 выбираем произвольно.

[1] Всюду в этом тексте под путём имеется в виду путь без повторяющихся рёбер

...

Во-вторых, все абзацы, начинающиеся со слов «докажем, что» лучше оформить отдельными утверждениями/предложениями/леммами. В таком виде рассуждение значительно легче читается. А при необходимости сослаться на доказанный факт, это опять-таки в таком виде сделать проще.

Доказательство утверждения 1. (Версия 3.2)

...

Лемма 1. В результате применения описанного алгоритма граф G не перестанет быть деревом.

Доказательство леммы 1.

...

Утверждение 2.

Число i, упоминаемое на шаге 4 алгоритма, действительно существует.

Доказательство утверждения 2.

...

Утверждение 3.

Описанный алгоритм закончит свою работу.

Доказательство утверждения 3.

...

Замечание 1.

В результате производимых перестроек число рёбер в графе не изменяется. При этом, если алгоритм совершил в общей сложности k шагов, то граф G_k и будет искомой ломаной.

Ну и наконец вернёмся к самой формулировке утверждения 1. Пока мы писали его доказательство, мы поняли, что ломаная как таковая не нужна. Значит, следуя принципу бритвы Оккама, её упоминание вообще нужно исключить. Но тогда становится непонятно, в каких терминах вообще его сформулировать? Описание нашего крутейшего алгоритма появится только в доказательстве; в терминах последовательности вершин тоже как-то не очень понятно звучит...

Кажется, утверждение 1 нам в таком виде вообще не нужно — всё равно его доказательство содержит практически всё решение задачи. Кроме того в процессе описания доказательства мы поняли, что последовательность вершин нужна только во вспомогательных целях, а существенная часть работы идёт с промежуточными графами G_k. Поэтому имеет смысл их ввести сразу на этапе описания алгоритма. Скорректируем исходный план работ:

- *формулируем алгоритм перестройки графа;*
- *доказываем, что в процессе его работы дерево остаётся деревом;*
- *доказываем, что алгоритм определён корректно и вообще завершается (утверждения 2 и 3 по версии 3.2);*
- *делаем вывод о том, сколько должно быть рёбер в G.*

На следующей странице приведён текст, получившийся в итоге. Его и признаем окончательной версией решения.

А пока ещё раз подытожим общие принципы, которым нужно следовать:

1. *Если текст достаточно велик, он должен быть разбит на логические блоки, которые можно осмысливать по-отдельности.*
2. *Основная идея должна прослеживаться. В любой момент читатель должен понимать, к чему и зачем всё движется. При необходимости можно в начале текста привести общий план доказательства с перечнем основных логических шагов и их взаимосвязью.*
3. *Не нужно вводить новые понятия и обозначения, если без них можно обойтись без ущерба читаемости текста. И наоборот: зачастую бывает проще ввести новое обозначение/понятие, чем повторять описательное определение несколько раз по тексту.*
4. *Лучше не использовать длинных сложноподчинённых конструкций. Если помимо нетривиальной математики читателю придётся продираться ещё и через нетривиальную грамматику, он точно утомится.*

Обозначения.
G — исходное дерево
V — множество вершин G
$n = |V|$
Ответ.
Граф G имеет $n - 1$ ребро.
Решение.
Опишем алгоритм последовательного преобразования графа G. Обозначим через G_k тот граф, который получается из G после применения k шагов алгоритма (в частности, $G_0 = G$). Множеством вершин графа G_k также будет являться V.

Как вспомогательный объект будем строить последовательность $(v_1, v_2, \ldots)$ вершин из V.

1. Вершину v_1 выберем произвольно. Граф не изменяется ($G_1 = G_0$).
2. Если в какой-нибудь момент все вершины из V уже включены в последовательность, алгоритм завершается. Обозначим число проделанных к этому моменту шагов через K.
3. *(Удлинение)* Предположим, что на некотором шаге уже построена последовательность $(v_1, \ldots, v_k)$. Пусть найдётся вершина v, соединённая ребром с v_k и ещё не включённая в последовательность. Тогда возьмём $v_{k+1} = v$. Граф не изменяется ($G_{k+1} = G_k$).
4. *(Перестройка)* Предположим, что на некотором шаге уже построена последовательность $(v_1, \ldots, v_k)$. Пусть теперь не найдётся вершины, соединённой ребром с v_k и ещё не включённой в последовательность. Выберем наибольшее число $i < k$ со следующим свойством: найдётся вершина v, соединённая ребром с v_i и ещё не включённая в последовательность. Существование такого i будет доказано ниже в утверждении 1. Берём $v_{k+1} = v$. Множеством рёбер графа G_{k+1} будет являться множество рёбер графа G_k за вычетом ребра (v_i, v_{k+1}) и с добавлением (v_k, v_{k+1}).

Замечание 1.
Для любого $k < K$ число рёбер в графах G_k и G_{k+1} одинаково.
Лемма 1.
Все получаемые графы G_k являются деревьями.
Доказательство леммы 1.
Доказательство будем вести индукцией по k.
База:
В силу $G_1 = G$ база очевидна.
Шаг индукции:
Мы предполагаем, что G_k — дерево.
Если $(k+1)$-ый шаг — удлинение, то $G_{k+1} = G_k$. Значит, G_{k+1} — дерево.
Пусть $(k+1)$-ый шаг — перестройка. Рассмотрим какой-нибудь путь[2] P в графе G_k. Поставим ему в соответствие путь $\alpha(P)$ в графе G_{k+1} следующим образом:

- Если в P не встретилось ребро (v_{k+1}, v_i) — то самое, которое было удалено на $(k+1)$-ом шаге алгоритма, — берём $\alpha(P) = P$.
- Если в P встретилось ребро (v_{k+1}, v_i), заменим его на последовательность рёбер $(v_{k+1}, v_k), (v_k, v_{k-1}), \ldots, (v_{i+1}, v_i)$. Полученный путь возьмём в качестве $\alpha(P)$.
- Если в P встретилось ребро (v_i, v_{k+1}), заменим его на последовательность рёбер $(v_i, v_{i+1}), \ldots, (v_{k-1}, v_k), (v_k, v_{k+1})$. Полученный путь возьмём в качестве $\alpha(P)$.

И наоборот: рассмотрим какой-нибудь путь Q в графе G_{k+1}. Поставим ему в соответствие путь $\beta(Q)$ в графе G_k следующим образом:

- Если в Q не встретилось ребро (v_{k+1}, v_k) — то самое, которое было добавлено на $(k+1)$-ом шаге алгоритма, — берём $\beta(Q) = Q$.

[2]Всюду в этом тексте под путём имеется в виду путь без повторяющихся рёбер

- Если в Q встретилось ребро (v_{k+1}, v_k), то за ним должны следовать рёбра $(v_k, v_{k-1}), \ldots, (v_{i+1}, v_i)$ — это следует из выбора i и из того, что G_k — дерево. Заменим всю последовательность этих рёбер $(v_{k+1}, v_k), (v_k, v_{k-1}), \ldots, (v_{i+1}, v_i)$ на одно ребро (v_{k+1}, v_i). Полученный путь возьмём в качестве $\beta(Q)$.
- Аналогично, если в Q встретилось ребро (v_k, v_{k+1}), перед ним следуют рёбра $(v_i, v_{i+1}), \ldots, (v_{k-1}, v_k)$. Заменим их все на одно ребро (v_i, v_{k+1}). Полученный путь возьмём в качестве $\beta(Q)$.

Легко видеть, что отображения α и β взаимно обратны, поэтому каждое из них взаимно-однозначно. Кроме того, концы пути при отображениях α и β сохраняются. Значит, две вершины связаны единственным путём в графе G_{k+1} тогда и только тогда, когда они связаны единственным путём в графе G_k. Из этого следует, что G_{k+1} — дерево.

Лемма 1 доказана.

Утверждение 1.

Число i, упоминаемое на шаге 4 алгоритма, действительно существует.

Доказательство утверждения 1.

Предположим, что это не так. Значит, в графе G_k ни одна из вершин, уже включённых в последовательность, не имеет соседей не из последовательности. А поскольку граф G_k связен (согласно лемме 1), получаем, что все вершины G уже включены в последовательность. Значит, алгоритм должен был уже завершиться ($k = K$ и шаг 4 выполняться не должен). Противоречие.

Утверждение 2.

Алгоритм закончит свою работу.

Доказательство утверждения 2.

Действительно, на каждом шаге алгоритма число вершин в последовательности возрастает на 1 и при этом вершины в последовательности не повторяются. Поскольку число вершин в графе G конечно, алгоритм завершится.

Утверждение 3.

Число рёбер в графе G_K равно $n - 1$.

Доказательство утверждения 3.

Из построения графа G_K следует, что у него есть рёбра $(v_1, v_2), \ldots, (v_{n-1}, v_n)$. Поскольку G_K — дерево, никаких других рёбер у него нет. Значит, всего рёбер у него $n - 1$.

Решение исходной задачи

Из утверждения 3 и замечания 1 следует, что число рёбер графа G равно $n - 1$.
