САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Олимпиада школьников СПбГУ по математическому моделированию и искусственному интеллекту

Примеры заданий отборочного этапа 2022/2023 учебный год

9-11 классы

Задача 1: Поезд без расписания (20 баллов)

Автор задачи: Олег Басков
Автор разбора: Олег Басков

Формулировка (Вариант 1)

На цифровой железной дороге действует жесткое расписание, в котором для каждого поезда указаны коды станций, через которые он идет, время прибытия и время отправления. Если соответствующее время не имеет смысла, ставится прочерк. Между любыми двумя станциями в каждый момент времени может находиться не более одного поезда. Станции имеют достаточно путей, чтобы вместить все поезда. Если две станции соединены железной дорогой, то по ней обязательно ходит хотя бы один поезд. Все поезда движутся с одинаковой скоростью. Начальник железной дороги хочет устроить внезапную проверку работы станции 010. Для этого он планирует выехать со станции 001 на служебном поезде не ранее 10:00.

В какое самое раннее время начальник сможет прибыть на станцию 010 так, чтобы не нарушить график движения других поездов?

В качестве ответа надо ввести только время прибытия на станцию 010. Ответ необходимо ввести в формате ЧЧ:ММ (через двоеточие). Например, 12:31 или 09:02

Расписание:

001 - 10:28
002 10:46 10:46
005 11:02 11:05
008 11:21 11:23
010 11:42 11:42
009 11:49 -

009 - 09:58
007 10:05 10:06
005 10:12 10:13
003 10:18 10:21
002 10:33 10:46
001 11:04 -

001 - 11:04
002 11:22 11:23
005 11:39 11:40
007 11:46 11:46
006 12:03 12:06
004 12:18 12:20
001 12:26 -

001 - 10:09
002 10:27 10:29
005 10:45 10:46
007 10:52 10:55
009 11:02 11:05
010 11:12 -

010 - 09:47
007 09:55 10:05
009 10:12 10:12
006 10:21 10:21
003 10:28 10:33
002 10:45 11:22
001 11:40 -

001 - 09:45
003 10:03 10:03
006 10:10 10:12
007 10:29 10:30
010 10:38 10:41
009 10:48 -

009 - 10:39
006 10:48 10:51
007 11:08 11:11
005 11:17 11:19
003 11:24 11:25
001 11:43 -

001 - 09:43
004 09:49 09:49
006 10:01 10:01
009 10:10 10:13
007 10:20 -

007 - 09:51
005 09:57 09:58
003 10:03 10:05
002 10:17 11:40
001 11:58 -

001 - 10:16

004 10:22 10:23
006 10:35 11:08
007 11:25 11:27
005 11:33 11:33
008 11:49 11:49
010 12:08 -

010 - 09:59
009 10:06 10:20
007 10:27 10:29
005 10:35 10:35
003 10:40 10:42
001 11:00 -

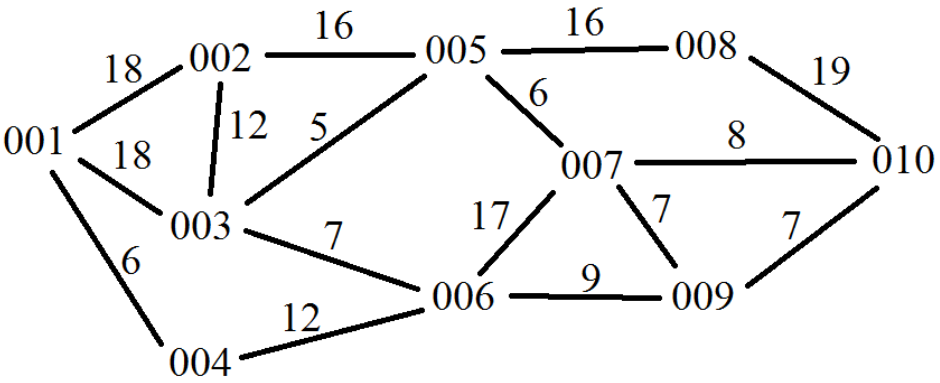
001 - 09:51
004 09:57 10:01
006 10:13 10:29
007 10:46 10:46
010 10:54 10:57
009 11:04 -

009 - 10:22
006 10:31 11:25
007 11:42 11:46
005 11:52 11:53
002 12:09 12:10
001 12:28 -

Решение

Начнем с составления графа железных дорог: вершинами будут станции, а ребра будут ставиться, если станции соединены перегоном. По условию задачи станции соединены тогда и только тогда, когда хотя бы один поезд едет от одной станции к другой. Пройдем по расписанию первого поезда: его маршрут 001 -- 002 -- 005 -- 008 -- 010 -- 009, поэтому добавляем ребра вдоль этого маршрута в граф. Обработав все расписание, получим следующий результат.

Out[ ]=



На каждом ребре подпишем время хода в минутах, рассчитанное по расписанию. Например, первый поезд отправляется со станции 001 в 10:28 и прибывает на станцию 002 в 10:46, что означает, что время в пути между станциями 001 и 002 составляет 18 минут. Аналогично рассчитываются остальные значения.

Для дальнейших вычислений удобно также сформировать график занятости перегонов. Для каждого ребра на графе укажем промежутки времени, в которые они заняты поездом. Для этого еще раз просматриваем расписание каждого поезда и время его нахождения на перегоне заносим в список.

Перегон 001 -- 002: 10:09-10:27, 10:28-10:46, 10:46-11:04, 11:04-11:22, 11:22-11:40, 11:40-11:58, 12:10-12:28.

Перегон 001 -- 003: 09:45-10:03, 10:42-11:00, 11:25-11:43.

Перегон 001 -- 004: 09:43-09:49, 09:51-09:57, 10:16-10:22, 12:20-12:26.

Перегон 002 -- 003: 10:05-10:17, 10:21-10:33, 10:33-10:45.

Перегон 002 -- 005: 10:29-10:45, 10:46-11:02, 11:23-11:39, 11:53-12:09.

Перегон 003 -- 005: 09:58-10:03, 10:13-10:18, 10:35-10:40, 11:19-11:24.

Перегон 003 -- 006: 10:03-10:10, 10:21-10:28.

Перегон 004 -- 006: 09:49-10:01, 10:01-10:13, 10:23-10:35, 12:06-12:18.

Перегон 005 -- 007: 09:51-09:57, 10:06-10:12, 10:29-10:35, 10:46-10:52, 11:11-11:17, 11:27-11:33, 11:40-11:46, 11:46-11:52.

Перегон 005 -- 008: 11:05-11:21, 11:33-11:49.

Перегон 006 -- 007: 10:12-10:29, 10:29-10:46, 10:51-11:08, 11:08-11:25, 11:25-11:42, 11:46-12:03.

Перегон 006 -- 009: 10:01-10:10, 10:12-10:21, 10:22-10:31, 10:39-10:48.

Перегон 007 -- 009: 09:58-10:05, 10:05-10:12, 10:13-10:20, 10:20-10:27, 10:55-11:02.

Перегон 007 -- 010: 09:47-09:55, 10:30-10:38, 10:46-10:54.

Перегон 008 -- 010: 11:23-11:42, 11:49-12:08.

Перегон 009 -- 010: 09:59-10:06, 10:41-10:48, 10:57-11:04, 11:05-11:12, 11:42-11:49.

Теперь для решения задачи применим алгоритм Дейкстры. Начинаем со станции отправления 001 и перебираем все возможные пути к соседним станциям. Движение по перегону 001 -- 002 занимает 18 минут, и такое окно в расписании занятости этого перегона появится лишь в 12:28, так что самое раннее время прибытия на станцию 002 таким способом составит 12:44. Перегон 001 -- 003 также занимает 18 минут, и он будет свободен в 10:03, так что на станции 003 можно оказаться в 10:21. Перегон 001 -- 004 занимает 6 минут, и он свободен уже в 10:00, так что на станции 004 можно оказаться в 10:06.

Будем отслеживать самое раннее возможное время прибытия на каждую станцию. Сейчас это: 004 -- 10:06, 003 -- 10:21, 002 -- 12:44.

Теперь берем станцию, в которую можно добраться раньше всего --- это 004 в 10:06. Перебираем все возможные варианты продолжения маршрута из нее. Он всего один: перегон 004 -- 006, время в пути 12 минут. Сверяясь с расписанием занятости перегона, обнаруживаем, что ближайшая возможность проехать по нему будет в 10:35, и на станции 006 можно оказаться в 10:47.

Таким образом, времена прибытия на станции, которые мы еще не рассматривали, такие: 003 -- 10:21, 006 -- 10:47, 002 -- 12:44.

Ближайшая по времени станция 003, берем ее. Рассматриваем маршруты в 002, 005 и 006, поскольку возврат в 001 явно не является оптимальным способом добраться до цели. Перегон 002 -- 003 занимается 12 минут, и соответствующее окно будет в 10:45, так что в 002 можно оказаться в 10:57. Это лучше, чем найденный ранее вариант прямого проезда 001 -- 002, так что обновляем самое раннее время прибытия в 002. Далее, перегон 003 -- 005 занимает 5 минут, и это можно сделать сразу с 10:21, так что в 005 можно оказаться в 10:26. Наконец, перегон 003 -- 006 занимает 7 минут, но придется подождать до 10:28, и в 006 получится прибыть лишь в 10:35. Однако это лучше, чем ранее найденное время, так что обновляем рекорд.

Получается, что времена прибытия в еще не рассмотренные станции стали такими: 005 -- 10:26, 006 -- 10:35, 002 -- 10:57.

Продолжаем построение маршрута со станции 005, поскольку до нее можно добраться быстрее. Перегон 005 -- 002 занимает 16 минут, и до 11:02 он будет занят, так что попасть на станцию 002 раньше, чем как в текущем рекорде в 10:57, не получится. Станция 003 уже рассмотрена, в нее мы точно не доберемся раньше, чем в уже найденные 10:21. Так что рассматриваем вариант 005

-- 007, перегон со временем хода 6 минут, и его можно будет занять в 10:35, так что на станции 007 можно будет оказаться в 10:41. И вариант 005 -- 008: перегон на 16 минут можно занимать сразу в 10:26, и на станции 008 можно оказаться в 10:42.

Таким образом, текущие рекорды времени прибытия: 006 -- 10:35, 007 -- 10:41, 008 -- 10:42, 002 -- 10:57.

Рассматриваем станцию 006. Перегон 006 -- 007 на 17 минут можно будет занять лишь в 12:03, но это не улучшит уже найденное время прибытия в 007. Перегон 006 -- 009 на 9 минут можно занять в 10:48 и прибыть на станцию 009 в 10:57.

Продолжаем отслеживать времена прибытия: 007 -- 10:41, 008 -- 10:42, 002 -- 10:57, 009 -- 10:57.

Рассматриваем станцию 007. Путь 007 -- 009 займет 7 минут, и он свободен в 10:41, так что в 009 можно прибыть в 10:48. Это лучше, чем ранее найденный вариант, так что обновляем рекорд. Путь 007 -- 010 займет 8 минут, однако выехать в 10:41 нельзя, поскольку в 10:46 перегон будет занят другим поездом. Так что самое раннее время отправления по этому маршруту будет в 10:54, и в 010 можно прибыть в 11:02.

Обновленные времена прибытия: 008 -- 10:42, 002 -- 10:57, 009 -- 10:48, 010 -- 11:02.

Рассматриваем станцию 008 и единственный маршрут вперед 008 -- 010, занимающий 19 минут. В 10:42 перегон свободен, так что можно прибыть на 010 в 11:01. Обновляем рекорд.

Времена прибытия: 002 -- 10:57, 009 -- 10:48, 010 -- 11:01.

Станцию 002 рассматривать поздно, поскольку во все соседние станции можно прибыть раньше, чем в 10:57. Так что продолжаем со станции 009. Из нее до 010 7 минут, и перегон как раз освобождается в 10:48, так что можно ехать сразу и в 010 оказаться в 10:55. Это рекорд.

Таким образом, ответ: самое раннее возможное время прибытия на станцию 010 -- 10:55.

Формулировка (Вариант 2)

На цифровой железной дороге действует жесткое расписание, в котором для каждого поезда указаны коды станций, через которые он идет, время прибытия и время отправления. Если соответствующее время не имеет смысла, ставится прочерк. Между любыми двумя станциями в каждый момент времени может находиться не более одного поезда. Станции имеют достаточно путей, чтобы вместить все поезда. Если две станции соединены железной дорогой, то по ней обязательно ходит хотя бы один поезд. Все поезда движутся с одинаковой скоростью. Начальник железной дороги хочет устроить внезапную проверку работы станции 010. Для этого он планирует выехать со станции 001 на служебном поезде не ранее 08:00.

В какое самое раннее время начальник сможет прибыть на станцию 010 так, чтобы не нарушить график движения других поездов?

В качестве ответа надо ввести только время прибытия на станцию 010. Ответ необходимо ввести в формате ЧЧ:ММ (через двоеточие). Например, 12:31 или 09:16

Расписание:

001 - 08:30
002 08:35 08:37
005 08:54 08:54
008 09:11 09:11
010 09:23 09:25
009 09:42 -

009 - 08:04
007 08:15 08:17
008 08:24 08:24
005 08:41 08:54
002 09:11 09:11
001 09:16 -

001 - 08:19
002 08:24 09:11
005 09:28 09:29
007 09:42 09:44
006 09:50 09:52
004 10:11 10:12
001 10:17 -

001 - 08:37
002 08:42 09:28
005 09:45 09:46
007 09:59 10:02
008 10:09 10:09
010 10:21 -

010 - 07:49
007 07:58 08:00
006 08:06 08:07
003 08:26 08:26
002 08:35 08:42
001 08:47 -

001 - 08:38
003 08:45 08:45
006 09:04 09:04
007 09:10 09:13
010 09:22 09:42
009 09:59 -

009 - 08:29
007 08:40 08:40
006 08:46 09:04
003 09:23 09:25
004 09:40 09:41
001 09:46 -

001 - 08:09
002 08:14 08:14
003 08:23 08:25
004 08:40 08:43
006 09:02 09:10
007 09:16 -

007 - 08:28
005 08:41 09:45
002 10:02 10:02
003 10:11 10:12
001 10:19 10:21
004 10:26 10:28
006 10:47 10:48
007 10:54 -

001 - 08:14
004 08:19 08:19
006 08:38 08:46
007 08:52 08:52
005 09:05 09:11
008 09:28 09:31
010 09:43 -

010 - 08:29
009 08:46 08:48
007 08:59 09:05
005 09:18 10:02
002 10:19 10:21
001 10:26 -

001 - 07:48
004 07:53 07:55
006 08:14 08:14

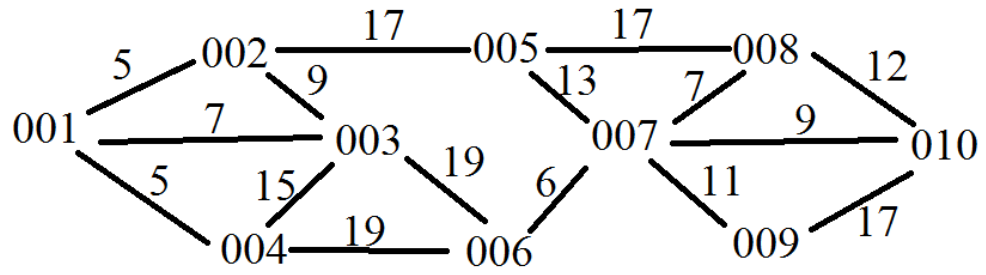
007 08:20 08:23
010 08:32 08:46
009 09:03 -

009 - 07:40
010 07:57 07:59
007 08:08 08:08
006 08:14 09:02
004 09:21 09:24
001 09:29 -

Решение

Начнем с составления графа железных дорог: вершинами будут станции, а ребра будут ставиться, если станции соединены перегоном. По условию задачи станции соединены тогда и только тогда, когда хотя бы один поезд едет от одной станции к другой. Пройдем по расписанию каждого поезда и добавим ребра вдоль его маршрута в граф. Получим следующий результат.

Out[ ]=



На каждом ребре подпишем время хода в минутах, рассчитанное по расписанию. Например, первый поезд отправляется со станции 001 в 08:30 и прибывает на станцию 002 в 08:35, что означает, что время в пути между станциями 001 и 002 составляет 5 минут. Аналогично рассчитываются остальные значения.

Для дальнейших вычислений удобно также сформировать график занятости перегонов. Для каждого ребра на графе укажем промежутки времени, в которые они заняты поездом. Для этого еще раз просматриваем расписание каждого поезда и время его нахождения на перегоне заносим в список.

Перегон 001 -- 002: 08:09-08:14, 08:19-08:24, 08:30-08:35, 08:37-08:42, 08:42-08:47, 09:11-09:16, 10:21-10:26.

Перегон 001 -- 003: 08:38-08:45, 10:12-10:19.

Перегон 001 -- 004: 07:48-07:53, 08:14-08:19, 09:24-09:29, 09:41-09:46, 10:12-10:17, 10:21-10:26.

Перегон 002 -- 003: 08:14-08:23, 08:26-08:35, 10:02-10:11.

Перегон 002 -- 005: 08:37-08:54, 08:52-09:11, 09:11-09:28, 09:28-09:45, 09:45-10:02, 10:02-10:19.

Перегон 003 -- 004: 08:25-08:40, 09:25-09:40.

Перегон 003 -- 006: 08:07-08:26, 08:45-09:04, 09:04-09:23.

Перегон 004 -- 006: 07:55-08:14, 08:19-08:38, 08:43-09:02, 09:02-09:21, 09:52-10:11, 10:28-10:47.

Перегон 005 -- 007: 08:28-08:41, 08:52-09:05, 09:05-09:18, 09:29-09:42, 09:46-09:59.

Перегон 005 -- 008: 08:24-08:41, 08:54-09:11, 09:11-09:28.

Перегон 006 -- 007: 08:00-08:06, 08:08-08:14, 08:14-08:20, 08:40-08:46, 08:46-08:52, 09:04-09:10, 09:10-09:16, 09:44-09:50, 10:48-10:54.

Перегон 007 -- 008: 08:17-08:24, 10:02-10:09.

Перегон 007 -- 009: 08:04-08:15, 08:29-08:40, 08:48-08:59.

Перегон 007 -- 010: 07:49-07:58, 07:59-08:08, 08:23-08:32, 09:13-09:22.

Перегон 008 -- 010: 09:11-09:23, 09:31-09:43, 10:09-10:21.

Перегон 009 -- 010: 07:40-07:57, 08:29-08:46, 08:46-09:03, 09:25-09:42, 09:42-09:59.

Теперь для решения задачи применим алгоритм Дейкстры. Начинаем со станции отправления 001 и перебираем все возможные пути к соседним станциям. Движение по перегону 001 -- 002 занимает 5 минут, и такое окно в расписании имеется сразу же в 08:00, так что самое раннее время прибытия на станцию 002 составит 08:05. Перегон 001 -- 003 занимает 7 минут, и он тоже свободен сразу в 08:00, так что на станции 003 можно оказаться в 08:07. Перегон 001 -- 004 занимает 5 минут, и он тоже свободен в 08:00, так что на станции 004 можно оказаться в 08:05.

Будем отслеживать самое раннее возможное время прибытия на каждую станцию. Сейчас это: 002 -- 08:05, 004 -- 08:05, 003 -- 08:07.

Теперь берем станцию, в которую можно добраться раньше всего, например, 002 в 08:05. Перебираем все возможные варианты продолжения маршрута из нее. Перегон 002 -- 003 занимает 9 минут, и как раз можно успеть прибыть в 003 в 08:14, пока перегон свободен. Однако мы ранее нашли возможность оказаться в 003 уже в 08:07, так что вариант маршрута 001 -- 002 -- 003 отбрасываем как неоптимальный. Перегон 002 -- 005 занимает 19 минут, и тоже можно успеть проехать по нему до первого поезда с расписанием, так что на станцию 005 можно прибыть в 08:24.

Текущие рекорды прибытия на еще не рассмотренные станции: 004 -- 08:05, 003 -- 08:07, 005 -- 08:24.

Рассматриваем станцию 004. До станции 003 за две минуты не добраться, так что рекорд прибытия туда не побить. Остается вариант 004 -- 006, перегон со временем хода 19 минут,

который, однако, освободится лишь в 09:21, так что на станцию 006 таким маршрутом можно прибыть лишь в 09:40.

Продолжаем отслеживать рекорды: 003 -- 08:07, 005 -- 08:24, 006~-- 09:40.

Берем станцию 003 с самым ранним временем прибытия. Возвращаться в 002 или 004 смысла нет, поэтому рассматриваем перегон 003 -- 006. Он занимает 19 минут, и, как видно из графика занятости перегонов, как раз можно, отправившись в 08:26, прибыть на станцию 006 в 08:45. Это лучше, чем ранее полученное время, так что обновляем рекорды: 005 -- 08:24, 006 -- 08:45.

Рассматриваем варианты продолжения маршрута через 005. Перегон 005 -- 007 занимает 13 минут, и проскочить между графиковыми поездами не получится, только в 09:59 можно будет отправиться и оказаться на станции 007 в 10:12. Перегон 005 -- 008 занимает 17 минут, и тоже отправиться получится лишь в 09:28, так что прибытие на 008 таким маршрутом состоится самое раннее в 09:45.

Текущие рекорды: 006 -- 08:45, 008 -- 09:45, 007 -- 10:12.

Теперь берем станцию 006, из которой вперед ведет только перегон 006 -- 007. Путь по нему занимает 6 минут, и по графику занятости перегонов видно, что можно отправляться в 08:52 и прибывать в 007 в 08:58. Это лучше, чем ранее найденное время, так что обновляем рекорды: 007 -- 08:58, 008 -- 09:45.

Рассматриваем 007. Перегон 007 -- 008 занимает 7 минут, и, отправившись в 08:58, можно прибыть на станцию 008 в 09:05. Это лучше, чем ранее найденное время. Далее, перегон 007 -- 009 занимает 11 минут, и можно отправиться в 08:59 и оказаться на станции 009 в 09:10. Наконец, перегон 007 -- 010 занимает 9 минут, как раз свободен в 08:58, так что прибыть на нужную станцию 010 можно в 09:07.

Текущие рекорды: 008 -- 09:05, 010 -- 09:07, 009 -- 09:10.

Осталось увидеть, что со станции 008 на станцию 010 за две минуты не попасть, так что рекорд в 09:07 побит не будет, и это время является ответом.

Задача 2: План поездки (30 баллов)

Автор задачи: Александр Кривошеин
Автор разбора: Александр Кривошеин

Формулировка (Вариант 1)

Оля впервые приехала в Санкт-Петербург, но уже выписала себе список достопримечательностей, которые она хотела бы посетить, указав для каждой достопримечательности степень интереса в баллах, денежные затраты и затраты по времени.

Оля пробудет в Санкт-Петербурге **три полных дня**. Помогите ей составить план максимально интересный план посещения достопримечательностей и уложиться при этом в **бюджет 4000р.**

Место	Баллы	Время (части дня)	Стоимость (руб.)
Петергоф: фонтаны, парки, дворцы	10	$\frac{3}{5}$	800
Русский музей	7	$\frac{1}{2}$	700
Петропавловский собор	6	$\frac{1}{3}$	300
Пушкин: парк, дворец	8	$\frac{2}{3}$	1000
Прогулка по Невскому проспекту	5	$\frac{1}{4}$	200
Кронштадт: Остров фортов	9	$\frac{2}{3}$	900
Парк аттракционов: Диво остров	7	$\frac{2}{5}$	1200
Зоопарк	8	$\frac{2}{3}$	600
Исаакиевский собор	5	$\frac{1}{5}$	400
Экскурсия на катере по Неве	6	$\frac{1}{3}$	600

В качестве ответа введите целое число, равное суммарному числу баллов интереса мест из составленного плана.

Например, если план состоит из Зоопарка и Русского музея, то суммарное число баллов интереса равно 15.

Решение

Обозначим:

баллы за b_i
время за t_i
стоимость за c_i

$i = 1, \ ..., 10.$

Пусть составлен некоторый план поездки. Составим бинарный вектор для этого плана $(D_1, \ ..., D_{10})$, где

$D_i = 1$, если i-ое место попало в план

$D_i = 0$, если i-ое место не попало в план.

Составленный план назовём подходящим, если он удовлетворяет условиям

$$t_1 D_1 + ... + t_{10} D_{10} \le 3,$$

$$c_1 D_1 + ... + c_{10} D_{10} \le 4000.$$

Таким образом, среди подходящих планов надо найти набор с максимальной величиной баллов

$$b_1 D_1 + ... + b_{10} D_{10} \ \to \ \max$$

Простейший способ получить решение заключается в реализации следующего алгоритма:

1. Сгенерировать все возможные планы, то есть возможные бинарные вектора $(D_1, \ ..., D_{10})$. Ясно, что число таких векторов равно 2^{10} или 1024 вектора.
2. Среди всех возможных векторов выбрать те, которые являются подходящими
3. Выбрать среди подходящих векторов тот, который даёт максимальную величину баллов.

Для примера, приведён код в Wolfram Mathematica, реализующий этот алгоритм.

```
In[•]:= scores = {10, 7, 6, 8, 5, 9, 7, 8, 5, 6};
        times = {3/5, 1/2, 1/3, 2/3, 1/4, 2/3, 2/5, 2/3, 1/5, 1/3};
        costs = {800, 700, 300, 1000, 200, 900, 1200, 600, 400, 600};
        allVectors = Tuples[{0, 1}, 10];

        isCorrect[vector_] := (vector.times ≤ 3) && (vector.costs ≤ 4000);

        correctVectors = Select[allVectors, isCorrect];
        Max[ correctVectors.scores]

Out[•]= 48
```

Ответ: 48 баллов.

Формулировка (Вариант 2)

Оля впервые приехала в Санкт-Петербург, но уже выписала себе список достопримечательностей, которые она хотела бы посетить, указав для каждой достопримечательности степень интереса в баллах, денежные затраты и затраты по времени.

Оля пробудет в Санкт-Петербурге **три полных дня**. Помогите ей составить план максимально интересный план посещения достопримечательностей и уложиться при этом в **бюджет 3500р.**

Место	Баллы	Время (части дня)	Стоимость (руб.)
Петергоф: фонтаны, парки, дворцы	10	$\frac{2}{3}$	1000
Русский музей	7	$\frac{1}{2}$	700
Петропавловский собор	6	$\frac{1}{3}$	300
Пушкин: парк, дворец	8	$\frac{2}{3}$	800
Прогулка по Невскому проспекту	5	$\frac{1}{4}$	200
Кронштадт: Остров фортов	9	$\frac{2}{3}$	900
Парк аттракционов: Диво остров	8	$\frac{1}{2}$	900
Зоопарк	8	$\frac{2}{3}$	600
Исаакиевский собор	5	$\frac{1}{5}$	400
Экскурсия на катере по Неве	6	$\frac{1}{4}$	600

В качестве ответа введите целое число, равное суммарному числу баллов интереса мест из составленного плана.

Например, если план состоит из Зоопарка и Русского музея, то суммарное число баллов интереса равно 15.

Решение

Аналогичным образом, применяем код в Wolfram Mathematica, реализующий алгоритм, описанный выше.

```
In[•]:= scores = {10, 7, 6, 8, 5, 9, 8, 8, 5, 6};
        times = {2/3, 1/2, 1/3, 2/3, 1/4, 2/3, 1/2, 2/5, 1/5, 1/4};
        costs = {1000, 700, 300, 800, 200, 900, 900, 600, 400, 600};
        allVectors = Tuples[{0, 1}, 10];

        isCorrect[vector_] := (vector.times ≤ 3) && (vector.costs ≤ 3500);

        correctVectors = Select[allVectors, isCorrect];
        Max[ correctVectors.scores]

Out[•]= 43
```

Ответ: 43 балла.

Задача 3: Модель данных (50 баллов)

Авторы задачи: Герасим Кривовичев
Автор разбора: Александр Кривошеин

Формулировка (Вариант 1)

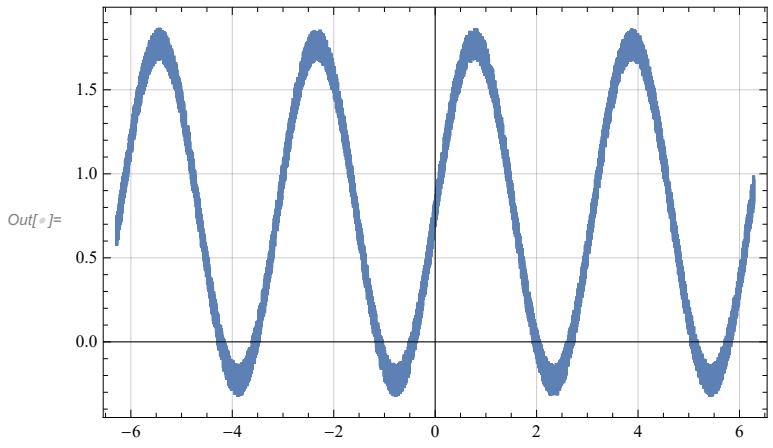
Петя прошёл курс по обучению искусственных нейронных сетей и решил применить свои знания. Задача в том, чтобы построить модель для неизвестной зависимости $y = f(x)$ по набору данных в виде пар чисел (x_i, y_i), $i = 1, \ldots, N$. Однако, его друг, взглянув на график данных, понял, что такая сложная модель не нужна, а можно использовать одну из тригонометрических функций и два параметра.

Помогите Пете подобрать зависимость вида $y = \varphi(x, a, b)$ и параметры a, b, которые приближают неизвестную зависимость $y = f(x)$ с наименьшей квадратичной ошибкой на данных, то есть следующая функция имеет наименьшее значение

$$I(a, b) = \sum_{k=1}^{N} (y_k - \varphi(x_k, a, b))^2.$$

В качестве ответа запишите величину ошибки $I(a, b)$ с точностью до 2 знаков после запятой. Например, 67,32

Данные находятся в файле “TaskDataVar1.xls” и представляют собой массив размера $10^4 \times 2$ из чисел. В первом столбце содержатся x_i, во втором y_i. График данных представлен ниже.



Файл можно скачать по ссылке:
https://disk.yandex.ru/i/gjrR8MpGdZs9lQ

Решение

Сначала надо определить модель для данных. Как указано в формулировке, это одна из тригонометрических функций и два параметра. График действительно похож на график зашумлённой тригонометрической функции
В нуле график пересекает ось Oy на уровне значения 0.7. График колеблется примерно от -0.3 до 1.8, значит есть сдвиг значений примерно на 0.7 вверх и соответствующая функция является

синусом. График задан на интервале [-6.28,6.28], то есть от -2π до 2π. На этом интервале на графике наблюдается примерно 4 полных колебания. То есть синус сжат примерно в 2 раза. Таким образом, наиболее подходящая зависимость имеет вид

$$y = \varphi(x, a, b) = \sin(a\,x) + b.$$

Далее, можно действовать несколькими способами. Первый заключается в построении достаточно плотной сетки параметров a и b. Для каждой пары параметров (a_i, b_i) из сетки можно вычислить значение ошибки

$$I(a, b) = \sum_{k=1}^{N} (y_k - \varphi(x_k, a, b))^2$$

и затем найти ту пару, которая даёт минимальное значение ошибки $I(a, b)$.

Для примера, приведён код в Wolfram Mathematica, реализующий этот алгоритм.

```
impData = Import["D:\\TaskDataVar1.xls", "Data"][[1]];
X = impData[[All, 1]];
y = impData[[All, 2]];
MyModel[x_, a_, b_] := Sin[a x] + b
Error[X_, y_, a_, b_] := Sum[(y[[k]] - MyModel[X[[k]], a, b])^2, {k, 1, 10000}]
```

In[•]:=
```
Parameters = Flatten[ Table[{a, b}, {a, 1.9, 2.1, 0.01}, {b, 0.7, 0.8, 0.01}], 1];
AllErrors = Error[X, y, #[[1]], #[[2]]] & /@ Parameters;
Min[AllErrors]
```

Out[•]=
```
33.6765
```

На выбранной сетке значение минимума оказалось равным 33.6765. Этот ответ близок к точному, но можно сделать ещё лучше. Один способов — это выбрать ещё более плотную сетку около тех значений параметров, которые дают значение ошибки, равное 33.6765.

Второй способ — это использовать метод градиентного спуска. Суть его в том, чтобы последовательно сдвигать вектор параметров (a, b) на малый по длине вектор, направленный в сторону противоположную вектору градиента функции ошибки. Градиент — это вектор из частных производных функции ошибки по параметру a и по параметру b:

$$\operatorname{grad} I(a, b) = \left(\frac{\partial I}{\partial a}, \frac{\partial I}{\partial b}\right), \text{ где}$$

$$\frac{\partial I}{\partial a} = \sum_{k=1}^{N} 2\,(y_k - \sin(a\,x_k) - b)\,\left(-\cos(a\,x_k)\right)\,x_k$$

$$\frac{\partial I}{\partial b} = \sum_{k=1}^{N} 2\,(y_k - \sin(a\,x_k) - b)\,(-1)$$

Метод градиентного спуска работает следующим образом:

1. Выбор начального значения параметров (a_0, b_0)
2. Найти значение градиента для этого набора: $\operatorname{grad} I(a_0, b_0)$
3. Задать малое значение $\alpha > 0$.
4. Найти новые значения параметров по формуле

$$(a_1, b_1) = (a_0, b_0) - \alpha \operatorname{grad} I(a_0, b_0).$$

При достаточно малом α значение функции ошибки на новых параметрах будет меньше значения этой функции на старых параметрах: $I(a_1, b_1) < I(a_0, b_0)$. Таким образом, алгоритм имеет вид:

1. Выбор начального значения параметров $(a_{\text{old}}, b_{\text{old}}) = (a_0, b_0)$
2. Повторять в цикле:

$$(a_{\text{new}}, b_{\text{new}}) := (a_{\text{old}}, b_{\text{old}}) - \alpha \operatorname{grad} I(a_{\text{old}}, b_{\text{old}})$$
$$(a_{\text{old}}, b_{\text{old}}) := (a_{\text{new}}, b_{\text{new}}).$$

Выбор параметра α отвечающего за величину сдвига влияет на сходимость метода к минимуму функции $I(a, b)$.

Для примера, приведён код в Wolfram Mathematica, реализующий этот алгоритм.

Сначала напишем методы, считающие значения модели и вектора, обратного градиенту.

```
In[•]:= MyModel[x_, a_, b_] := Sin[a x] + b
        myAntiGrad[x_, y_, a_, b_] :=
         {2 (y - MyModel[x, a, b]) * Cos[a * x] * x, 2 (y - MyModel[x, a, b])}
```

Затем в цикле из 500000 итераций проделаем градиентный спуск:

```
In[•]:= {a, b} = {2, 1};

        For[i = 1, i ≤ 500 000, i++,
          datInst = RandomChoice[impData];
          {a, b} = {a, b} + 0.0005 * myAntiGrad[datInst[[1]], datInst[[2]], a, b];
         ];
        Error[impData[[All, 1]], impData[[All, 2]], a, b]

Out[•]= 33.5868
```

Результат получился лучше, однако он не стабилен, при нескольких запусках будут получены немного различные ответы. Результат будет стабильнее, если сдвигать параметры на усреднённое значение из 100 градиентов. Это так называемый пакетный градиентный спуск. Также будем снижать размер параметра α с ростом числа итераций, например по правилу $\alpha = \frac{1}{10 + i}$, где i номер итерации. Тогда хватит и 5000 итераций.

```
In[•]:= {a, b} = {2, 1};
        For[i = 1, i ≤ 5000, i++,
          datInst = RandomChoice[impData, 100];
          {a, b} =
           {a, b} + 1/(10 + i) 0.01 * Sum[myAntiGrad[datInst[[k, 1]], datInst[[k, 2]], a, b],
              {k, 1, 100}];
         ];
        Error[impData[[All, 1]], impData[[All, 2]], a, b]

Out[•]= 33.3975
```

Ответ 33.40 ± 0.02 оценивался в наибольшее число баллов за эту задачу.

Формулировка (Вариант 2)

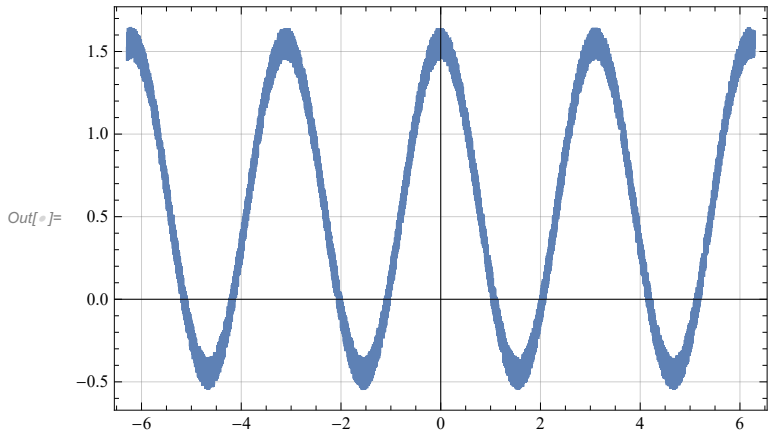
Петя прошёл курс по обучению искусственных нейронных сетей и решил применить свои знания. Задача в том, чтобы построить модель для неизвестной зависимости $y = f(x)$ по набору данных в виде пар чисел (x_i, y_i), $i = 1, \ldots, N$. Однако, его друг, взглянув на график данных, понял, что такая сложная модель не нужна, а можно использовать элементарную функцию и два параметра.

Помогите Пете подобрать зависимость вида $y = \varphi(x, a, b)$ и параметры a, b, которые приближают неизвестную зависимость $y = f(x)$ с наименьшей квадратичной ошибкой на данных, то есть следующая функция имеет наименьшее значение

$$I(a, b) = \sum_{k=1}^{N} (y_k - \varphi(x_k, a, b))^2.$$

В качестве ответа запишите величину ошибки $I(a, b)$ с точностью до 2 знаков после запятой. Например, 67,32

Данные находятся в файле “TaskDataVar2.xls” и представляют собой массив размера $10^4 \times 2$ из чисел. В первом столбце содержатся x_i, во втором y_i. График данных представлен ниже.



Файл можно скачать по ссылке:
https://disk.yandex.ru/i/VnLNcAtoC97R2w

Решение

Сначала надо определить модель для данных. Как указано в формулировке, это одна из тригонометрических функций и два параметра. График действительно похож на график зашумлённой тригонометрической функции

В нуле график пересекает ось Oy на уровне значения 1.5. График колеблется примерно от -0.5 до 1.5, значит есть сдвиг значений примерно на 0.5 вверх и соответствующая функция является косинусом. График задан на интервале [-6.28,6.28], то есть от -2π до 2π. На этом интервале на графике наблюдается примерно 4 полных колебания. То есть косинус сжат примерно в 2 раза. Таким образом, наиболее подходящая зависимость имеет вид

$$y = \varphi(x, a, b) = \cos(a\,x) + b.$$

```
In[•]:= impData = Import["D:\\TaskDataVar2.xls", "Data"][[1]];
        X = impData[[All, 1]];
        y = impData[[All, 2]];
        MyModel[x_, a_, b_] := Sin[a x] + b
        Error[X_, y_, a_, b_] := Sum[(y[[k]] - MyModel[X[[k]], a, b])^2, {k, 1, 10000}]
```

Применим сразу пакетный градиентный спуск, описанный выше. Градиент имеет вид

$$\text{grad}\, I(a, b) = \left(\frac{\partial I}{\partial a}, \frac{\partial I}{\partial b}\right), \text{ где}$$

$$\frac{\partial I}{\partial a} = \sum_{k=1}^{N} 2\,(y_k - \cos(a\,x_k) - b)\ \sin(a\,x_k)\,x_k$$

$$\frac{\partial I}{\partial b} = \sum_{k=1}^{N} 2\,(y_k - \cos(a\,x_k) - b)\ (-1)$$

```
In[•]:= MyModel[x_, a_, b_] := Cos[a x] + b
        myAntiGrad[x_, y_, a_, b_] :=
         {2 (y - MyModel[x, a, b]) * (-Sin[a * x]) * x, 2 (y - MyModel[x, a, b])}
```

Тогда

```
In[•]:= {a, b} = {2, 0.5};
        For[i = 1, i ≤ 5000, i++,
          datInst = RandomChoice[impData, 100];
          {a, b} =
           {a, b} + 1/(10 + i) 0.01 * Sum[myAntiGrad[datInst[[k, 1]], datInst[[k, 2]], a, b],
              {k, 1, 100}];
         ];
        Error[impData[[All, 1]], impData[[All, 2]], a, b]

Out[•]= 33.5836
```

Ответ 33.58 ± 0.02 оценивался в наибольшее число баллов за эту задачу.