Задачи, предлагаемые для летней стажировки в Яндексе

Содержание

1 Стажировка в группе разработки инструментов аналитики (Санкт-Петербург)

Поиск — это основной сервис Яндекса, каждый день им пользуются несколько миллионов человек. Любое изменение в поиске должно учитывать их интересы. Решения о совершенствовании алгоритмов поиска мы принимаем с учётом анализа статистики, мнений пользователей, своей интуиции, а иногда — экспериментов.

Наша команда разрабатывает инструменты анализа и оценки качества поисковых систем. Мы пишем модуль большой библиотеки, которая позволяет нам получить ответы различных поисковых систем (например, Яндекс и Google).

Задание. Выберите одну из задач, на каждую из них не стоит тратить больше 4 часов. Предпочтительный язык программирования — `java`. Решение присылайте в виде архива или ссылкой на репозиторий.

- **Парсер.** Есть Яндекс и к нему можно задавать запросы. При заполнении формы пользователь попадает на страницу результатов. Напишите код, который выделит из этой странице подокументно ответ системы, где документ это урл, текст заголовка, краткая аннотация и зелёная строчка. После этого, нужно выкачать

html-страницы этих документов и сохранить их в папочку в виде файлов. Аккуратно, Яндекс банит тех, кто много качает.

- **FixedSizeCache.** Есть такая структура данных, в которую можно складывать по ключу объекты, и количество таких пар ключ-значение ограничено, назовем такую структуру кэшом. Доступ к данным в кэше идет быстрее, чем выборка исходных данных из медленного источника (например, из базы данных) или их перевычисление, за счет чего уменьшается среднее время доступа к элементу. Но кэш ограничен, следовательно возникнет ситуация, когда следующий объект мы уже не сможем добавить в кэш, не удалив какой-нибудь элемент. Существует несколько стратегий выбора элемента, подлежащих замене. Среди известных: FIFO и LRU. Нужно спроектировать интерфейсы, реализовать структуру.

2 Стажировка в группе обработки мета-данных (Москва)

Вакансия предполагает анализ почтовых логов, поиск закономерностей, отклонения от них, поиск возможных багов.

Задание. Пусть у нас на диске есть файл размером 4 гигабайта. Его можно представить, как 2^{30} 32-битных беззнаковых чисел. Нужно отсортировать этот файл. То есть программа должна сгенерировать ещё один файл размером в 4 гигабайта, в котором содержатся те же числа, что и в исходном, но упорядочены по возрастанию. Стандартные алгоритмы сортировки (`qsort`, `std::sort`) напрямую применить невозможно, так как для их выполнения нужно как минимум 4 гигабайта оперативной памяти. Но отсортировать числа на диске возможно, если использовать дополнительное пространство на жёстком диске.

Нужно написать консольную программу, которая в `argv[1]` получает имя файла, который нужно отсортировать (размер файла до 16Gb, размер кратен четырём), в `argv[2]` — имя файла, в который нужно записать отсортированные данные. Предпочтительный язык программирования — C++. Решение присылайте в виде архива или ссылкой на репозиторий. Ограничения:

1. Программа не может рассчитывать, что возможно выделить более 256Mb памяти.
2. Программа должна эффективно использовать несколько ядер.

Нам бы хотелось получить решение, по которому можно было бы оценить качество кода (простота чтения, скорость работы). Засчитывается решение, которое:

1. Компилируется ;)
2. Правильно завершается, даже при неправильных данных, недостаточных ресурсах, а не просто падает в случайных местах.

3. Корректно сортирует произвольный файл.

4. Код можно прочитать и понять.

5. Работает эффективно. «Эффективно» значит:

   (a) используя немного памяти, зато полностью загружая CPU,

   (b) нельзя придумать существенно более эффективное (на десятки процентов, или с принципиально лучшей временной сложностью) решение. Экономию в 2-3% путём микрооптимизаций не учитываем.

3 Стажировка в в службе разработки речевых продуктов (Москва)

Яндекс уделяет большое внимание речевым технологиям. Ввод данных голосом является достаточно естественным, поэтому в последнее время все больше продуктов появляется использующих технологию распознавания речи. Хотя само направление относительно новое, но с ним мы связываем большие надежды. Распознавание речи является наукоёмкой, технически сложной задачей и в ней есть множество нетривиальных и интересных проблем. Задумок много, а возможности команды ограничены, поэтому нам нужны люди способные прорабатывать подобные идей. Работа будет интересна тем, кто любит challenge.

Нам нужен стажёр, который будет помогать разработке ключевых компонентов технологии распознавания речи. Работа потребует понимания технологии и возможно некоторое участия в научной деятельности. Технология будет ядром множества будущих продуктов и сервисов Яндекса, которыми будут пользоваться миллионы пользователей.

Задание. Дана скрытая Марковская модель.

1. 4 состояния: B, St1, St2, E.

2. Вероятности перехода:
   B $\rightarrow$ St1 0.526
   B $\rightarrow$ St2 0.474
   St1 $\rightarrow$ E 0.002
   St1 $\rightarrow$ St1 0.969
   St1 $\rightarrow$ St2 0.029
   St2 $\rightarrow$ E 0.002
   St2 $\rightarrow$ St1 0.063
   St2 $\rightarrow$ St2 0.935

3. Возможна эмиссия трёх символов: 'a', 'b', 'c'.

4. Эмиссионные вероятности:
   St1:
   'a', 0.005
   'b', 0.775
   'c', 0.220
   St2:
   'a', 0.604
   'b', 0.277
   'c', 0.119

5. Данные симуляции такой HMM приведены в файле hmmdata.

Требуется на языке C++:

1. Реализовать структуры данных для HMM.

2. Реализовать алгоритм Viterbi, предсказывающий наиболее вероятную последовательность состояний по данным эмиссии, оценить эффективность алгоритма по реальным данным о состояниях (рассчитать True Positives, False Positives, True Negatives, False Negatives и F-меру для задачи детекции состояния St1).

3. Реализовать алгоритм Forward-Backward, оценивающий вероятности состояний по данным эмиссии, оценить эффективность алгоритма по реальным данным о состояниях (рассчитать True Positives, False Positives, True Negatives, False Negatives и F-меру для задачи детекции состояния St1).

4 Стажировка в отделе теоретических и прикладных исследований (Москва)

Мы ище несколько человек в отдел для решения различных задач. Например, предлагаем провести исследования характера зарождения и распространения информации в сети. Желательно знание python и скриптовых языков и сильная математика, приветствуется знание C++. Основной акцент исследования будет приходится на изучение характера перехода людей на страницу — типы источников, кривая популярности страницы и построении классификаторов по этим данным. Другой вариант задачи: усовершенствование нового машинного обучения, желательно знать алгоритмы ML и программировать на C++. Так же есть задача фичей: есть пул данных с некоторым количеством факторов, хочется выбрать оптимальный поднабор фичей так, чтобы качество классификатора на нем было лучше (или не хуже), чем на всем наборе. Желательно знание python и алгоритмов ML.

Задание. Доктор Клуни и доктор Эдвардс в составе бригады скорой помощи выехали по вызову в один из отдалённых районов. Однако, когда диспетчер получил вызов, он успел записать только адрес дома и номер квартиры K1, а затем связь прервалась.

К счастью диспетчер вспомнил, что по этому же адресу дома некоторое время назад скорая помощь выезжала в квартиру K2, которая расположена в подъезде P2 на этаже N2.

Известно, что в доме M этажей и количество квартир на каждой лестничной площадке одинаково. Напишите программу, которая вычисляет номер подъезда P1 и номер этажа N1 квартиры K1. На вход программе подаются пять положительных целых чисел K1, M, K2, P2, N2. Все числа не превосходят 1000. Напечатайте два числа P1 и N1. Если входные данные не позволяют однозначно определить P1 или N1, вместо соответствующего числа напечатайте 0. Если входные данные противоречивы, напечатайте два числа -1. Предпочтительный язык программирования — `C++` (возможен и `python`). Решение присылайте в виде архива или ссылкой на репозиторий.

5 Стажировка в отделе оценки качества поиска (Санкт-Петербург)

Поиск и выделение схожих паттернов поведения пользователей, выделение в базе ЛПС групп пользователей со схожими параметрами и их детальный анализ. Требования: иметь представления о распределённых системах like MapReduce/Hadoop, умение работать с raw логами, знание какого-либо ЯП, который позволяет обрабатывать логи, знание статистики на уровне: постановка гипотезы и её проверка, представление о методах data mining — кластеризация, классификация и способах проверки качества конкретных реализаций методов.

Задания (нужно выполнить все).

1. Чтобы сравнить качество двух поисковых алгоритмов, был проведён эксперимент: замерены значения целевой метрики по случайной выборке из 20-ти поисковых запросов. Выборка фиксирована, запросы задаются в одинаковом порядке.

query1	0.3133	0.5379
query2	0.6353	0.5382
query3	0.4809	0.5384
query4	0.738	0.539
query5	0.5251	0.5397
query6	0.5145	0.5401
query7	0.5923	0.5406
query8	0.567	0.5448
query9	0.4146	0.5459
query10	0.3801	0.5534
query11	0.2808	0.5579
query12	0.4978	0.5583
query13	0.6634	0.561
query14	0.5227	0.5762
query15	0.4336	0.5769
query16	0.5269	0.5868
query17	0.5519	0.5875
query18	0.5368	0.6025
query19	0.4661	0.6059
query20	0.6516	0.6105

Можно ли по этим данным сделать вывод, что качество одного поискового алгоритма в среднем лучше, чем другого согласно выбранной целевой метрике? На каком уровне значимости?

2. Логи — это набор строк, где каждая строка представляет одну из возможных записей: метаданные следующей поисковой сессии, запрос, клик или переход на другую поисковую систему. Метаданные следующей сессии (`TypeOfRecord = M`):
`SessionID TypeOfRecord USERID`
Запрос (`TypeOfRecord = Q`):
`SessionID TimePassed TypeOfRecord SERPID QueryID ListOfURLs`
Клик (`TypeOfRecord = C`):
`SessionID TimePassed TypeOfRecord SERPID URLID`
`SessionID` — уникальный идентификатор пользовательской сессии.
`TypeOfRecord` — тип записи. Это может быть запрос (Q), клик (C), переход (S) либо метаданные следующей сессии (M).
`TimePassed` — время, прошедшее с начала текущей сессии в условных временных единицах.
`QueryID` — уникальный идентификатор текста запроса.
`SERPID` — идентификатор поисковой выдачи, уникальный на уровне сессии.
`URLID` — уникальный идентификатор документа.
`ListOfURLs` — список документов, отранжированный слева направо в том порядке, в каком они были показаны пользователям на странице выдачи Яндекса (сверху вниз).

Строки про конкретную сессию идут подряд и отсортированы по времени. Приведите код, который принимал бы на вход файл с логами, а на выходе выдавал бы файл с сессиями в виде последовательности действий вида `sessionID QCCQC`. Можно пользоваться любым языком программирования или статистическим пакетом.

3. Ниже приводятся данные по ряду стран: страна, средний доход на душу населения, уровень грамотности, уровень детской смертности, продолжительность жизни. Выделить и характеризовать схожие группы стран. Указать используемые расстояния между объектами и кластерами. Обосновать выбор метода кластеризации и количества кластеров.

Brazil	10326	90	23.6	75.4
Germany	39650	99	4.08	79.4
Mozambique	830	38.7	95.9	42.1
Australia	43163	99	4.57	81.2
China	5300	90.9	23	73
Argentina	13308	97.2	13.4	75.3
United_Kingdom	34105	99	5.01	79.4
South_Africa	10600	82.4	44.8	49.3
Zambia	1000	68	92.7	42.4
Namibia	5249	85	42.3	52.9
Georgia	4200	100	17.36	71
Pakistan	3320	49.9	67.5	65.5
India	2972	61	55	64.7
Turkey	12888	88.7	27.5	71.8
Sweden	34735	99	3.2	80.9
Lithuania	19730	99.6	8.5	73
Greece	36983	96	5.34	79.5
Italy	26760	98.5	5.94	80
Japan	34099	99	3.2	82.6