Об авторе

Всем привет! Для тех, с кем мы не знакомы лично или виртуально - меня зовут Лариса, и я уже больше трех тел работаю в Google в Mountain View на позиции Software Engineer. За это время я успела провести 174 интервью - телефонных и очных. А также больше двух лет еженедельно позаседать в специальной комиссии, которая принимает решение о найме в Google, где я видела несчетное количество отзывов других на интервью. Также, еще до устройства в Google, я проходила польшое количество интервью в качестве кандидата в разные компании - с разной степенью успешности ;).

Так что про интервью я знаю где-то в спектре от “много” до “очень много”. И, самое главное, я знаю как эти интервью оцениваются, и что нанимающая команда ищет в кандидатах - из первых рук и большого количества личного опыта.

Мне можно написать на allagentis@gmail.com. Вот тут находится пост о проекте.

Введение

Когда-то давно я помогала ребятам готовиться к интервью, проводя тестовые интервью по Скайпу. Со временем количество желающих существенно превысило мои возможности, и интервью я делать перестала.

Вместе с тем, я считаю, что для успешного прохождения интервью критичными являются 2 фактора - подготовка, и практика интервью. Когда я должна была проходить свои интервью, будучи на стажировке (для перехода на полную ставку нужно пройти два интервью вместо пяти), я тренировалась постоянно, прося коллег меня проинтервьюировать. И практика принесла свои плоды - я получила оффер.

Но еще больший “прорыв” в сознании у меня случился, когда я начала интервьюировать сама. Опыт интервьюирования открывает совершенно новый пласт в осознание того, что в интервью важно, на что обращает внимание интервьюер, и как он потом интервью оценивает. Опыт интервьюирования помогает понять:

- Как мыслит интервьюер, чего он ждет от кандидата
- Как интервьюер готовится к интервью
- Как интервьюер воспринимает ответы кандидата, что важно сказать, а что не очень важно
- Что лучше сказать, а о чем лучше не говорить

Согласитесь, все это очень ценные навыки для кандидата. Поэтому если у вас есть возможность и знания побыть не только кандидатом, но и интервьюером, я бы рекомендовала этой возможностью воспользоваться - это мега-ценный опыт.

Я постаралась разделить участников так, чтобы интервьюер был полее опытным, чем кандидат. Но, как и в реальных интервью, может оказаться, что кандидат знает в какой-то области больше, чем интервьюер. Если окажется, что это ваша ситуация, то я очень советую вам воспользоваться этой возможностью и потренероваться интервьюированию с интервьюером, который в чем-то знаем меньше, чем вы.

В Google такие ситуации тоже бывают (и у меня бывали), и очень важно пройти интервью хорошо даже при таких входных данных. Если у интервьюера останется ощущение, что кандидат зазнался (пусть даже и по делу), то ни к чему хорошему это не приведет - ведь один из параметров, по которому оцениваются интервью, это насколько с кандидатом комфортно общаться. Никто не хочет работать с букой или зазнайкой, каким бы умным он не был.

Логистика

Кандидаты: Через пару дней я пришлю вам контакт вашего интервьюера. Ваша задача с ним связаться и договориться о взаимно-удобном времени. Также приложите к письму ваше резюме, чтобы интервьюер примерно знал, на каком уровне задавать вопросы.

Интервьюеры: Если вдруг ваш кандидат не пришлет резюме, то попросите его прислать. Во-первых, резюме это очень важный фактор для подготовки к интервью. Не стесняйтесь ставить планку выше для тех, у кого много лет опыта работы (5+) и ждать, что они пишут код быстрее чем те, кто только закончил универ. Именно так делает Google и другие компании. Во-вторых, если вам есть что сказать про резюме, то не стесняйтесь дать кандидату совет по его улучшению (по email, не во время интервью).

Все:

1. Все мы тут интеллигентные люди, поэтому наверняка напоминание излишне, но давайте будем друг с другом вежливы. И кандидата, и интервьюера объединяет общая цель - помочь друг другу подготовиться к интервью.
2. Процесс запланирован так, чтобы большинство людей смогло побыть и интервьюером, и кандидатом.
3. Лично я предпочитаю проводить интервью через Skype. Место, где кандидаты пишут код - Google Docs. Я заранее создаю документ, и присылаю кандидатам емаил с ссылкой. Если решите воспользоваться Google docs, не забудьте предоставить кандидату право в документ писать. Если вы предпочитаете другие средства связи, или написания кода - пожалуйста, это непринципиально, на самом деле.
4. Постарайтесь не сильно затягивать назначение интервью. Думаю, месяца вполне хватит, чтобы договориться, и проинтервьюироваться, и проинтервьюировать. Если проект покажет себя с хорошей стороны и вы захотите продолжить тренироваться, то накапливать интервью - не самая продуктивная идея.
5. Кандидат должен писать код прямо в Google docs (или что вы там используете). Во время реального интервью кандидату не разрешать пользоваться Eclipse, чтобы проверить программу.

Анализ кандидата

Вот, на что я смотрю, когда получаю резюме кандидата для интервью:

1) **Опыт работы**. Если кандидат опытный (5+ лет), то я жду, что он умеет хорошо решать простые задачи, и в целом его код не стыдно читать. То есть если кандидат пишет что-то вроде (на C++):

```
void my_func(int x) {
  int xx = x/2;
  …
  return xxxx;
```

```
}
```

то это явный повод насторожиться.

Кстати, я обычно не учитываю опыт работы во время учебы (рекрутеры и комиссия его тоже не учитывают, если что). Или если учитываю, то с коэффициентом ½. Кандидат, который учился и работал одновременно, не равен кандидату, который все это время только работал.

2) **Уровень образования**. От тех, кто окончил PhD я жду более глубокого взгляда на вещи, что ли. На умение анализировать проблему с разных сторон. И если PhD кандидат дает односложные и поверхностные ответы в области, в которой он вроде как провел много лет, то это тоже повод насторожиться.

3) **Основной язык программирования**. Если кандидат последние 5 лет писал на Java, то было бы не очень честно спрашивать у него детали управления памятью в C++, и наоборот. Также как и вопросы, которые ориентированы на специфику не-Java. Например, “напишите memcpy” - не очень хороший вопрос для Java кандидата.

4) **Нет ли у кандидата явных странностей в резюме**. Например, за последние 3 года он сменил 5 мест работы. Если они есть, я вежливо, но поинтересуюсь почему так. Велика вероятность, что за частой сменой работы кроется неуживчивость, или кандидат недостаточно хорош и его быстро увольняли (впрочем, это бы всплыло в интервью).

Кстати, однажды у меня был такой кандидат. У него было 4 смены работы за 3 года, и он интервьюировался в Google. По его признанию, с первого места работы он ушел сам, так как ему там не понравилось, следующие 2 были маленькие стартапы, которые закрылись через пару месяцев после того, как он туда пришел. Последнее место работы тоже выглядит как скоро закрывающееся, и он просто устал от вечно крахующих стартапов, и хочет найти наконец нормальное место работы.

В общем, человек явно не умел выбирать себе стартапы для работы. Но в целом причина уважительная, и это его неумение никак не помешало бы ему работать в Google. Так что помехой для рассмотрения его кандидатуры это не стало.

5) **Прочие умения**. Я, например, вполне сносно знаю Haskell. И если я вижу у кандидата в резюме Haskell в позиции “хорошо знаю”, то я могу задать несложный вопрос на эту тему (на 2-3 минуты, так как моя главная цель в интервью проверить другие навыки, не Haskell). Пару раз оказывалось, что кандидат даже не особо представлял, что это такое, хотя уверенно писал об этом в резюме. В общем, так делать не надо.

И еще однажды у меня был кандидат, который утверждал, что у него оранжевый рейтинг на TopCoder. И который 30 минут писал IntToString, с багами. Я решила, что либо он

соврал, либо проходил интервью с серьезного бодуна - и то, и другое не очень хорошо, о чем я и написала в своем отзыве.

В любом случае, если вы видите в резюме, что кандидат профи в чем-то, в чем вы тоже хорошо разбираетесь, то можете задать и небольшой вопрос об этом тоже.

Структура интервью

Моя структура выглядит примерно так:

5 минут - Я представляюсь и прошу кандидата рассказать о чем-нибудь из его IT жизни. В общем, отвожу это время на нетехнические вопросы. Обычно я стараюсь не проводить больше 5 минут, разговаривая о резюме кандидата, так как красиво говорить о резюме - не критичный навык.

2-3 минуты - Очень простые вопросы за знания основ программирования в языке, который кандидат знает лучше всего. На уровне “Чем в Java отличается интерфейс от абстрактного класса?” или “Зачем нужны шаблоны в C++?”.

10-15 минут - Задача, на написание кода. Я специально подбираю несложные задачи, которые тестируют только написание кода, который я ожидаю кандидат может написать быстро. Никаких сложных алгоритмов, “а теперь напишите этот алгоритм, чтобы он работал на 100 машинах” и всего такого прочего. Вы удивитесь, сколько кандидатов не проходят этот простой тест (на телефонных интервью).

20-25 минут - Исходя из предположения, что кандидат справился с простой задачей на написание кода, я предлагаю ему задачу посложнее. Обычно, усложненную версию первой задачи, или совсем другую задачу. Тут я уже хочу, чтобы кандидат знал алгоримты, умел сравнить, например, hash map vs binary tree.

2-3 минуты - Последние пару минут я трачу на вопросы кандидатов. В нашем проекте мы их можем потратить на фидбек.

Итого: **45-50 минут**.

Важно: Планируйте интервью и вопросы так, чтобы оно длилось 45-50 минут, не больше. Это стандартная продолжительность Google интервью.

Темы интервью

Мы ориентируемся на эти темы:

- Основы (сложение/вычитание, двоичное исчисление, биты/байты…)
- Строки
- Массивы
- Тестирование кода

На каком уровне задавать вопросы нужно ориентироваться исключительно на кандидата. Опытные и махровые кандидаты должны без проблем решать сложные задачи, сравнивать разные алгоримические подходы, считать сложность алгоритмов. Им можно и нужно задавать задачи посложнее.

Для менее опытных кандидатов можно больше ориентироваться на написание хорошего кода без ошибок, чем на глубокое знание алгоритмов.

Кандидатам: Вы можете предупредить интервьюера, что вы не очень сильны в какой-то области.

Интервьюерам: Постарайтесь найти границу знаний кандидата. Какой бы уровень не был у вашего кандидата, постепенно усложняйте вопрос и остановитесь там, где кандидат уже не особо много знает. Если граница находится на “не особо хорошо пишет простой код”, то это сигнал, что кандидату придется долго и упорно готовиться, чтобы достичь уровня реального интервью.

Технические вопросы

На самом деле, материалов везде полно - Google вам в помощь. Внизу приведены те, которые я видела или знаю.

Теория:

1) Coursera (*en*) для тех, кто любит готовиться основательно, и у кого есть время
2) Кормен - основательно, но иногда длинновато.
3) Скиена - интересно, но надо уже что-то знать
4) Алгоритмы просто - не читала, но название обещает, что должно быть просто

Задачи:

1) Книга “Cracking the Coding Interview” (*en, ru*). Ее можно найти на торрентах. Она, кстати, была переведена на русский язык. Я ОЧЕНЬ советую основательно изучить ее в рамках подготовки. Поскольку тем у нас немного, то времени много занять оно не должно.
2) Сайт careercup.com (*en*)
3) Сайт geeksforgeeks.org (*en*)
4) Моя любимая книга (*ru*). Задачи начинаются с очень простых, и постепенно усложняются. Очень важно не перескакивать, и решать даже те, которые “элементарно, Ватсон”. Именно по этой книге я готовилась к своим интервью в Google на обе стажировки - успешно.

Конечно, это только верхушка айсберга. Материалов и задач можно найти намного больше.

Какие вопросы не задавать

Если вы интервьюер, то, пожалуйста, не задавайте кандидатам вопросы на совсем другие темы. Разве что у вас была предварительная договоренность расширить тематику.

Также ориентируйтесь на кандидата - если кандидат не особо силен в алгоритмах, то можно попросить его написать числа Фибоначчи (с рекурсией и без, и попросить сравнить достоинства и недостатки обоих), или IntToString.

Именно для того, чтобы определить примерный уровень кандидата, я вначале задаю достаточно простую задачу на код. И, в зависимости от того, насколько кандидат хорош, приспосабливаю свои последующие вопросы.

Как готовиться к интервью

1) Подготовьте несколько задач, которые вы будете спрашивать. Обычно у меня имеется 4-5 задач, на случай, если кандидат решает их очень быстро.

2) Я начинаю интервью с простой задачи на код. Примером такой задачи может быть:

Для менее опытных кандидатов:
- IntToString
- Fibonacci numbers
- Переведите число в двоичную систему исчисления
- Выведите все пары (x, y, z) так, чтобы x^2 + y^2 = z^2

Для более опытных кандидатов:

- Найдите наиболее часто встречающиеся число в строке
- Напишите функцию сжатия строки в следующем формате: “abba” -> “1a2b1a”

В зависимости от того, насколько быстро и качественно интервьюер решает простую задачу, я выбираю вторую задачу.

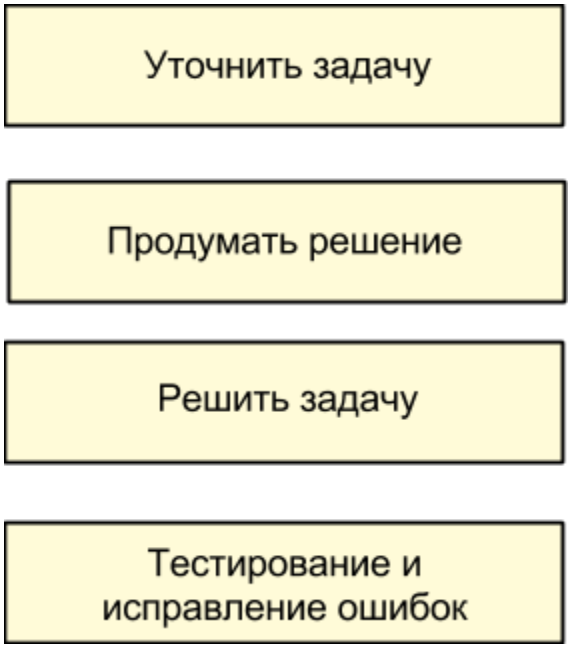
3) Более сложная задача обычно требует некоторых нетривиальных знаний:

- Опреледите, можно ли в данном массиве найти два числа с суммой N

Обязательно заранее изучите ваши задачи со всех сторон заранее. Какие есть подходы, какие из них лучше и почему. Какое решение самое быстрое (обычно искользует Hash).

Как оценивать технические вопросы

Вот примерный ход решения, которому по моему опыту следуют хорошие кандидаты:



1) **Уточнить задачу**. Это важный момент. Некоторые кандидаты бросаются писать код сразу (“Решить задачу”), и потом оказывается, что они не совсем правильно поняли задание и решали какую-то другую задачу. И даже если задание понято правильно, всегда есть открытые вопросы. Например:

- Для строк: “Ограничена ли строка по длине?” и “Можно ли ограничиться ASCII строкой?”
- Для чисел: “Нужно ли рассматривать целые числа только, или дробные тоже?”

- Для массивов: “Ограничен ли как-то размер массивов, или может быть, что в массиве будет несколько (десятков) миллионов элементов?

2) **Продумать решение**. Это хороший знак, если кандидат в двух словах опишет что он собирается сделать. На этом этапе можно обсудить ход решения, задать наводящие вопросы.

3) **Решить задачу.** На этапе написания кода я смотрю на:

- Насколько хороший код пишет кандидат в смысле эстетики. Не называет ли переменные абы как, проверяет ли входные параметры.
- Не зависает ли кандидат над тривиальными вещами - например написанием цикла
- Не делает ли кандидат тривиальных ошибок в языке программирования, который он должен бы знать хорошо
- Насколько быстро кандидат решает задачу. Если у него занимает 40 минут решить простую задачу, например Fibonacci numbers, то это не очень хорошо.
- Знает ли кандидат сложность его решения (O-нотация). Есть ли решение лучше, может ли кандидат оптимизировать свое решение (на уровне алгоритма).

4) **Тестирование и исправление ошибок**. Вот это тоже очень важно, однозначно на втором месте после собственно решения задачи. Хорошие кандидаты могут сделать ошибки в коде во время решение, и это ОК если они сами протестируют код, и найдут эти ошибки.
Когда кандидат говорит “Готово!”, даже не удосужившись протестировать код вручную на простых примерах, и в коде есть ошибки, я никогда не даю такому кандидату хорошую рекомендацию. Причем важно, чтобы кандидат не просто тестировал код, а тестировал его на краевых примерах тоже - 0, NULL, пустая строка итп.
Если кандидат код протестировал, но ошибок не нашел, то это вообще EPIC FAIL.

Нетехнические вопросы

Что я спрашиваю

1) Ваш любимый предмет в университете, и почему он вам нравится (у тех, кто недавно учился)
2) Какая область IT вам интересна, и почему. Например, базы данных или пользовательские интерфейсы (у средне опытных кандидатов).
3) Расскажите о проекте X (берется из резюме). Какие задачи вам показались сложными на этом проекте? Почему? (у более опытных кандидатов)
4) На каких позициях вы работали, и какие позиции вам были интереснее всего и почему? Тут подвох такой, что если кандидат интервьюируется на позицию, скажем, программиста, но больше всего ему было интересно заниматься разработкой веб-дизайна, то как минимум имеет смысл сказать кандидату, что позиции веб дизайнеров у нас тоже есть.

Что я не спрашиваю

1) Самый сложный баг, с которым вам приходилось сталкиваться.

Почему: Это вопрос больше на память и подготовку, которые вовсе не обязательно коррелируют с тем, что мы ищем в кандидатах.

2) Как вы решали сложную ситуацию с коллегой на работе.

Почему: Мой ответ - у меня когда-то давно был коллега, который принципиально терпеть не мог русских (белорусов, украинцев). Я эту ситуацию не решала и общалась с коллегой по-минимуму. Что это говорит обо мне? ИМХО, ничего - я могу быть как асоциалом, так и нормальным человеком.

Bottom line

Первые 5 минут мной используются исключительно для того, чтобы оценить насколько кандидат умеет внятно и структурно излагать свои мысли.

Какие вопросы не задавать

К нашему тестовому проекту это не особо относится, но обычно на интервью под табу попадают вопросы, на основании которых кандидат может решить, что его дискриминируют и пойти судиться с компанией. Например:

- Какая у вас сексуальная ориентация? или Пойдете ли вы на гей-парад завтра?
- Планируете ли вы детей в скором будущем?
- Какой страны у вас гражданство?
- А вам было сложно быть темнокожим в университете?

Отзыв на интервью

Вы вполне можете дать вашему кандидату отзыв в устной форме. Пожалуйста будьте вежливы и корректны, расскажите, что вам понравилось как кандидат сделал, что вам кажется кандидат должен проработать получше.

Тезисно

- Задайте вступительный вопрос (5 минут максимум), и 2-3 технических вопроса (40 минут). Первый технический вопрос должен быть простым, последующие вопросы адаптируйте к уровню кандидата
- Подготовьте и хорошенько изучите хотя бы 5 вопросов разной сложности, чтобы было из чего выбирать в процессе интервью
- Будьте вежливы, но не стесняйтесь прервать кандидата, если он долго отвечает на вопрос, или постепенно уходит от темы
- Дайте возможнсть кандидату самому решить задачи, но если надо, задавайте наводящие вопросы/подсказки
- Не задавайте вопросы на смекалку - aka puzzles. Они ничего не говорят о том, насколько кандидат хороший программист на самом деле.
- Если кандидат “застрял” на дольше, чем 1-2 минуты, то дайте подсказку или подсказки, помогите кандидату. Если не получается, переходите к другой задаче - может оказаться, что это просто случайный ступор.
- Если кандидат не решил ни одной задачи - вы задавали слишком сложные задачи, и ни кандидат, ни интервьюер не получили максимум пользы от такого интервью. Постарайтесь приспособить задачи и вопросы к уровню кандидата.
- Во время интервью основное время должен говорить кандидат, а не интервьюер
- Я советую подбирать вопросы на careercup.com или geeksforgeeks.org или в книге “Cracking the coding interview” (или ее русском аналоге). Зачастую там можно найти и решения.

FAQ

1. Что делать если кандидат застрял на вопросе, и не может ничего придумать?

1) Попробовать дать подсказку
2) Если подсказка (позсказки) не помогли, то дать другую задачу, возможно попроще.

2. Можно ли давать подсказки?

Можно, и зачастую нужно. Но тут важно, чтобы подсказки были не в стиле “Надо делать так”, а наводящими вопросами.

3. У меня есть вопрос, на который я не могу найти ответа

1. Напишите мне на allagentis@gmail.com ИЛИ
2. Задайте свой вопрос тут. Ответы на вопросы которые я упустила я буду дописывать в эту секцию.

4. Мой кандидат мне не написал/мой интервьюер пропал

Напишите мне на allagentis@gmail.com