Есть профиль требуемого кандидата из 10 навыков, ранжированных по важности:

1. Java = 10 балов

2. SQL = 9 балов

... прочие навыки

10. Python = 1 бал

И есть профили кандидатов :

Кандидат-1

1. Git = 10 балов

2. Java = 9 балов

... прочие навыки

10. SQL = 1 бал

Кандидат-2

1. Python = 10 балов

2. SQL = 9 балов

... прочие навыки

10. Java = 1 бал

Кандидат-3

1. Pascal = 10 балов

2. Java = 9 балов

... прочие навыки

10. SQL = 1 бал

и т.д.

Как определить какой кандидат лучше подходит требованиям ?  
  
  
Отличная, четко сформулированная задача! Теперь, когда навыки определены, мы можем дать точный ответ.

Для определения лучшего кандидата我们需要一个系统的方法来计算 "схожесть" профиля кандидата с эталонным профилем. Вот пошаговый алгоритм:

### Шаг 1: Определяем метрику для сравнения

Самая логичная метрика в данном случае — \*\*взвешенное скалярное произведение\*\* (Weighted Dot Product). Её смысл: мы наказываем за несовпадение по важным навыкам и меньше penalize за несовпадение по второстепенным.

\*\*Формула:\*\*

`Score = (Балл\_кандидата\_по\_навыку\_1 \* Балл\_требования\_по\_навыку\_1) + (Балл\_кандидата\_по\_навыку\_2 \* Балл\_требования\_по\_навыку\_2) + ...`

\*\*Чем выше итоговый Score, тем лучше кандидат соответствует требованиям.\*\*

### Шаг 2: Создаем "таблицу требований"

Вынесем требуемые навыки и их вес (важность) в отдельную таблицу. Это наш эталон.

| Навык | Вес (Важность) |

| :----- | :------------: |

| Java | 10 |

| SQL | 9 |

| ... | ... |

| Python | 1 |

\*Примечание: Для простоты расчета мы будем учитывать только те навыки, которые есть в требованиях. Навыки кандидата, которых нет в требованиях (например, Git, Pascal), игнорируются, так как их вес для нас равен 0.\*

### Шаг 3: Расчет для каждого кандидата

\*\*Кандидат-1\*\*

\* Java: 9 (у кандидата) \* 10 (требуется) = \*\*90\*\*

\* SQL: 1 \* 9 = \*\*9\*\*

\* Python: (предположим, у кандидата нет Python) -> 0 \* 1 = \*\*0\*\*

\* \*\*Итоговый Score: 90 + 9 + 0 = 99\*\*

\*\*Кандидат-2\*\*

\* Java: 1 \* 10 = \*\*10\*\*

\* SQL: 9 \* 9 = \*\*81\*\*

\* Python: 10 \* 1 = \*\*10\*\*

\* \*\*Итоговый Score: 10 + 81 + 10 = 101\*\*

\*\*Кандидат-3\*\*

\* Java: 9 \* 10 = \*\*90\*\*

\* SQL: 1 \* 9 = \*\*9\*\*

\* Python: (предположим, нет) -> 0 \* 1 = \*\*0\*\*

\* Pascal: (не в требованиях) -> игнорируется

\* \*\*Итоговый Score: 90 + 9 + 0 = 99\*\*

### Шаг 4: Анализ результатов и вывод

| Кандидат | Итоговый Score |

| :-------- | :------------: |

| \*\*Кандидат-2\*\* | \*\*101\*\* |

| Кандидат-1 | 99 |

| Кандидат-3 | 99 |

\*\*Вывод: Лучше всего подходит Кандидат-2.\*\*

### Почему это работает и что это значит?

\* \*\*Кандидат-1 и Кандидат-3\*\* набрали по 99 баллов. У них сильный Java (9 из 10), что очень ценно, но их слабый SQL (1 из 9) сильно подвел, так как SQL — второй по важности навык.

\* \*\*Кандидат-2\*\* идеально попал во второй по важности навык (SQL: 9 из 9). Его главный недостаток — слабый Java (1 из 10). Однако, с точки зрения метрики, его \*\*идеальное соответствие высокоприоритетному навыку (SQL) перевесило провал в самом главном навыке (Java)\*\*. Он набрал 101 балл против 99 у других.

\* Эта метрика показывает, что \*\*Кандидат-2 обладает более сбалансированным профилем\*\* с точки зрения требований вакансии.

### Важные нюансы и улучшения метода:

1. \*\*Пороговые значения (Thresholds):\*\* Метрика не учитывает, что Java — критически важный навык. Если Java обязателен, даже при идеальном соответствии всем остальным пунктам, кандидата с 1 баллом по Java можно сразу отсеить. На практике часто вводят "порог" по ключевым навыкам.

2. \*\*Нормализация:\*\* Максимально возможный балл в этой системе — `10\*10 + 9\*9 + ... + 1\*1`. Чтобы получить оценку в процентах, можно разделить Score кандидата на этот идеальный Score. Но это имеет смысл, только если кандидат должен иметь все навыки.

3. \*\*Учет отсутствующих навыков:\*\* В данном расчете отсутствие навыка (например, Python у Кандидата-1) не наказывается, так как этот навык маловажен. Если бы Python был важен, это было бы большим минусом.

\*\*Итоговый рекомендация:\*\* Используйте \*\*взвешенную сумму совпадений\*\* (Weighted Score). Это простой и эффективный метод. Но всегда дополняйте его \*\*фильтром по критическим навыкам\*\* (например, "Java must be >= 5").