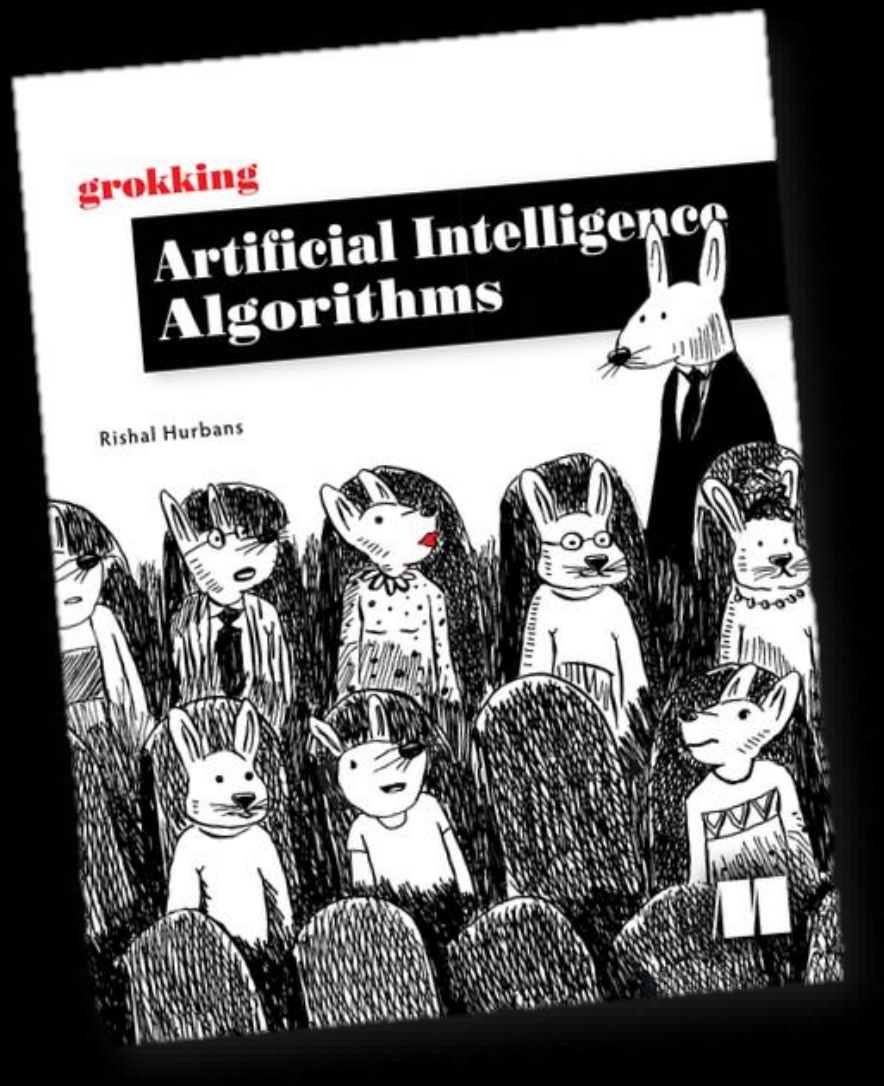


Grokking Algorithms

CHAPTER 7

في شابتر 7 هنتكلم عن Dijkstra's Algorithm والـ weighted graphs اللي بنستخدمها معاه .

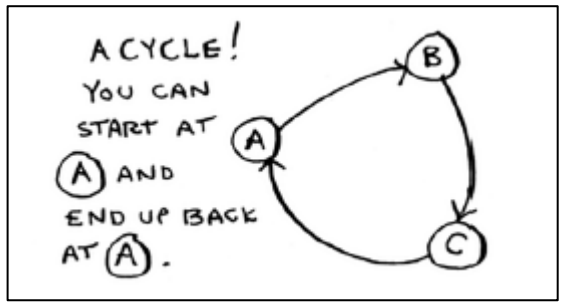
Dijkstra's Algorithm هوalgorithm بنستخدمه علشان نحسب أقصر مسار (shortest path) بين نقطتين في weighted graph .

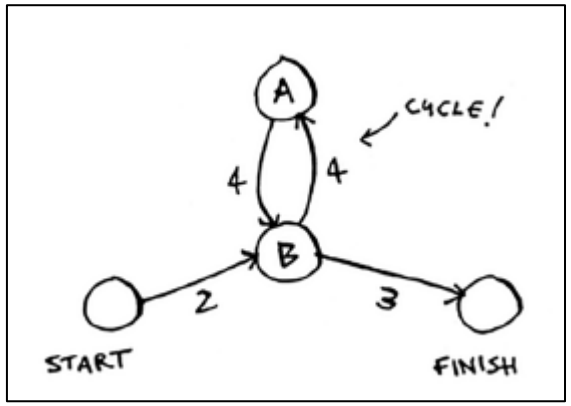
لما بنقول أقصر مسار، المقصود بيه المسار اللي أقل في الوزن (lowest weight) مش بالضرورة الأقصر في عدد الخطوات.

طيب، ليه مش بنستخدم breadth-first search هنا زي ما اتعلمنا في شابتر 6 الي فات ؟ لأن breadth-first search بيدور على المسار اللي فيه أقل عدد من الخطوات، لكن Dijkstra's Algorithm بيدور على المسار اللي وزنه أقل حتى لو كان فيه خطوات أكتر.

إزاي بتشتغل الخوارزمية دي؟ الفكرة ببساطة إننا بنبدأ من النقطة اللي عاوزين نوصلها ونسجل المسافات (الأوزان) لكل النقاط المتصلة بيها. بعد كده بنشوف أقصر مسار للنقاط اللي بعدها، وهكذا لحد ما نوصل للنقطة النهائية.

لازم ناخد بالنا من إن الـweighted graphs ممكن يكون فيها (cycle) دوايريعني ممكن تلف وترجع لنفس النقطة. وده ممكن يسبب مشكلة لو مشينا في cycle، علشان كده الخوارزمية دي بتتأكد إنها مش بتمشي في cycle.





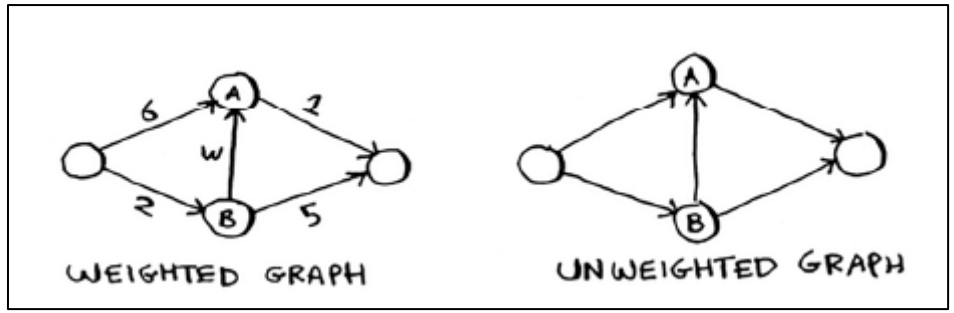
ايه المقصود بالاوزان هنا لاني سألت نفسي السؤال ده و انا بقرأ ؟

الأوزان هنا (weights) المقصود بيها قيمة أو تكلفة الربط بين نقطتين في الرسم البياني (graph) كل حافة (edge) بين نقطتين ممكن يكون لها وزن، والوزن ده بيمثل تكلفة الانتقال من نقطة للتانية .

مثلاً، لو عندك خريطة فيها مدن، وكل مدينة متصلة بمدينة تانية بطريق، ممكن تعتبر الأوزان هي المسافة بين المدن أو الوقت اللي هتاخده عشان توصل من مدينة للتانية. في الحالة دي، Dijkstra's Algorithm بيحاول يلاقي أقصر طريق بأقل وزن من نقطة البداية لنقطة النهاية.

يعني لو عندك ثلاث مدن A و B وC ، ولو المسافة بين A وB هي 5 كيلومتر، والمسافة بين B وC هي 2 كيلومتر، والمسافة بين A وC هي 10 كيلومتر، الأوزان هنا هتكون 5، 2، و10 على التوالي.

في weighted graph ، كل ما يكون الوزن أقل، كل ما يكون المسار أفضل من حيث التكلفة أو الوقت، ودور الخوارزمية هو إنها تلاقي المسار اللي فيه أقل وزن ممكن للوصول من نقطة لنقطة تانية.



Dijkstra's Algorithm works only with weighted graphs اللي مافيهاش أوزان سلبية no negative weights ، لأن وجود وزن سلبي ممكن يسبب مشاكل ويخلي الخوارزمية ما تشتغلش صح. عشان كده بنستخدم الخوارزمية دي مع directed acyclic graphs (DAGs) اللي مافيهاش دوائر.

في الخوارزميات اللي بتدور على أقصر مسار، في حاجة اسمها (Negative-Weight Edges) ، ودي بتخلي الوزن النهائي أقل من الصفر. خلينا ناخد مثال جديد ونرسمه عشان نفهم أكتر.

افترض إن عندنا المخطط ده:

- من A لـ B بيساوي 4
- من B لـ C بيساوي 1
- من C لـ D بيساوي -2
- من A لـ C بيساوي 3

دلوقتي، لو عايز توصل من A لـ D، هنلاحظ الآتي:

- المسار A -> B -> C -> D بيدينا الوزن الإجمالي 4 + 1 - 2 = 3
- المسار A -> C -> D بيدينا الوزن الإجمالي 3 + (-2) = 1

من الواضح إن المسار التاني هو الأقل وزناً، فطبيعي إنه يتختار. لكن لو استخدمت خوارزمية Dijkstra، هل هتختار المسار ده؟

خوارزمية Dijkstra وليه مش بتنفع مع الأوزان السالبة :

في خوارزمية Dijkstra ، بتبدأ من العقدة اللي انت عايز تبدأ منها وبتبني جدول فيه أقل وزن للوصول لكل عقدة. بس الخوارزمية دي بتفترض إن كل الأوزان موجبة، عشان كده مش بتاخد في الاعتبار الحواف اللي وزنها بالسالب. فلو في حافة سالبة، ده ممكن يدي نتائج مش دقيقة.

مثلاً، لو الخوارزمية بدأت من A، هتلاقي إن:

- A -> B = 4
- A -> C = 3

وبالتالي هتفترض إن C أقرب من B وهتروح على طول لـ C، وهتنسى إنك لو رحت لـ B وبعدها لـ C ممكن يديلك مسار أحسن.

ومن هنا بنستنتج إن **خوارزمية Dijkstra ما تنفعش مع الحواف اللي وزنها بالسالب.**

- بسبب العيب ده، محتاج تستخدم خوارزميات تانية زي Bellman-Ford للتعامل مع الحواف اللي وزنها بالسالب.

Recap

- Breadth-first search is used to calculate the shortest path for an unweighted graph.
- Dijkstra's algorithm is used to calculate the shortest path for a weighted graph.
- Dijkstra's algorithm works when all the weights are positive.
- If you have negative weights, use the Bellman-Ford algorithm.