**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**MÔN: PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN**

**ĐỀ TÀI: QUY HOẠCH ĐỘNG**

Giảng viên hướng dẫn: Thạc sĩ Nguyễn Thanh Sơn

Sinh viên thực hiện:

1. Đinh Duyên Bảo Duy MSSV: 18520658

2. Trần Minh Tiến MSSV: 18521492

3. Trần Tuấn Vỹ MSSV: 18520406

🙡🙢 Tp. Hồ Chí Minh, 12/2020 🙠🙣

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

*……., ngày……...tháng……năm 2020*

**Người nhận xét**

*(Ký tên và ghi rõ họ tên****)***

**MỤC LỤC**

[Chương 1: Tổng quan về Quy hoạch động 4](#_Toc60392496)

[1.1 Giới thiệu về quy hoạch động 4](#_Toc60392497)

[1.2 Ý tưởng thuật toán 5](#_Toc60392498)

[1.3 Các bước thược hiện quy hoạch động 5](#_Toc60392499)

[1.4 Các bước giải bài toán quy hoạch động 7](#_Toc60392500)

[1.5 Các hướng tiếp cận bài toán 7](#_Toc60392501)

[a/ Tabulation Method – Bottom Up 7](#_Toc60392502)

[b/ Memoization Method – Top Down 7](#_Toc60392503)

[c/ So sánh 2 cách tiếp cận: 7](#_Toc60392504)

[Chương 2: Các bài toán kinh điển về quy hoạch động 8](#_Toc60392505)

[2.1 Bài toán: 0/1 Knapsack 8](#_Toc60392506)

[a/ Bài toán: 8](#_Toc60392507)

[b/ Tóm tắt: 8](#_Toc60392508)

[c/ Hướng giải quyết: 9](#_Toc60392509)

[2.2 Bài toán dãy con đơn điệu dài nhất: 9](#_Toc60392510)

[a/ Bài toán: 9](#_Toc60392511)

[b/ Tóm tắt: 9](#_Toc60392512)

[c/ Hướng giải quyết: 10](#_Toc60392513)

[Chương 3: Phản ví dụ 10](#_Toc60392514)

[Chương 4: So sánh giữa quy hoạch động vào chia để trị 11](#_Toc60392515)

[Chương 5: Các ứng dụng của Quy hoạch động 11](#_Toc60392516)

# Tổng quan về Quy hoạch động

## Giới thiệu về quy hoạch động

* Thuật ngữ  “Dynamic Programming" ban đầu được sử dụng vào những năm 1940 bởi Richard Bellman. Ý nghĩa của thuật ngữ “Dynamic Programming” được thể hiện trong “Eye of the Hurricane: An Autobiography” được ông cho rằng ý nghĩa về cái tên không quá cao siêu so với nhiều người tưởng. (1984, trang 159).
* **Quy hoạch động** là một kĩ thuật thiết kế thuật toán theo kiểu chia bài toán lớn thành các bài toán con, sử dụng lời giải của các bài toán con để tìm lời giải cho bài toán ban đầu. Khác với chia để trị, quy hoạc động, thay vì gọi đệ quy, sẽ tìm lời giải của các bài toán con và lưu vào bộ nhớ (thường là một mảng), và sau đó lấy lời giải của bài toán con ở trong mảng đã tính trước để giải bài toán lớn. Việc lưu lại lời giải vào bộ nhớ khiến cho ta không phải tính lại lời giải của các bài toán con mỗi khi cần, do đó, tiết kiệm được thời gian tính toán
* Phương pháp này đã được áp dụng để giải hàng loạt bài toán thực tế trong các quá trình kỹ thuật cộng nghệ, tổ chức sản xuất, kế hoạch hoá kinh tế. Tuy nhiên cần lưu ý rằng có một số bài toán mà cách giải bằng quy hoạch động thì không thích hợp.
* Nguyên lý tối ưu của R.Bellmam được phát biểu như sau: “tối ưu bước thứ n bằng cách tối ưu tất cả con đường tiến đến bước n-1 và chọn con đường có tổng chi phí từ bước 1 đến bước n-1 và từ n-1 đến n là thấp nhất” (nhiều nhất).
* Nguyên lý tối ưu Bellman (cũng gọi là nguyên lý Bellman) có thể diễn giải theo một cách khác như sau: “Với mỗi quá trình điều khiển tối ưu, đối với trạng thái bắt đầu A0, với trạng thái A trong quá trình đó, phần quá trình kể từ trạng thái A xem như trạng thái bắt đầu cũng là tối ưu”.
* Thuật ngữ quy hoạch động nói lên thực chất của quá trình điều khiển là động: Có thể trong một số bước đầu tiên lựa chọn điều khiển tối ưu dường như không tốt nhưng tựu chung cả quá trình lại là tốt nhất.

## Ý tưởng thuật toán

* Chia bài toán lớn thành những bài toán con nhỏ hơn. Giải quyết bài toán con nhỏ, sau đó lưu kết quả với mục đích sử dụng lại để giải quyết bài toán tổng quát hơn.
* Khi giải một bài toán bằng cách “chia để trị” chuyển việc giải bài toán kích thước lớn về việc giải nhiều bài toán cùng kiểu có kích thước nhỏ hơn thì thuật toán này thường được thể hiện bằng các chương trình con đệ quy. Khi đó, trên thực tế, nhiều kết quả trung gian phải tính nhiều lần. Vậy ý tưởng cơ bản của quy hoạch động thật đơn giản: Tránh tính toán lại mọi thứ hai lần, mà lưu giữ kết quả đã tìm kiếm được vào một bảng làm giả thiết cho việc tìm kiếm những kết quả của trường hợp sau. Chúng ta sẽ làm đầy dần giá trị của bảng này bởi các kết quả của những trường hợp trước đã được giải. Kết quả cuối cùng chính là kết quả của bài toán cần giải. Nói cách khác phương pháp quy hoạch động đã thể hiện sức mạnh của nguyên lý chia để trị đến cao độ.

## Các bước thược hiện quy hoạch động

* Bước 1: Lập hệ thức
* Dựa vào nguyên lý tối ưu tìm cách chia quá trình giải bài toán thành từng giai đoạn, sau đó tìm hệ thức biểu diễn tương quan quyết định của bước đang xử lý với các bước đã xử lý trước đó. Hoặc tìm cách phân rã bài toán thành các “bài toán con” tương tự có kích thước nhỏ hơn, tìm hệ thức nêu quan hệ giữa kết quả bài toán kích thước đã cho với kết quả của các “bài toán con” cùng kiểu có kích thước nhỏ hơn của nó nhằm xây dựng phương trình truy toán (dạng hàm hoặc thủ tục đệ quy).
* Về một cách xây dựng phương trình truy toán: Ta chia việc giải bài toán thành n giai đoạn. Mỗi giai đoạn i có trạng thái ban đầu là t(i) và chịu tác động điều khiển d(i) sẽ biến thành trạng thái tiếp theo t(i+1) của giai đoạn i+1 (i=1,2,…,n-1). Theo nguyên lý tối ưu của Bellman thì việc tối ưu giai đoạn cuối cùng không làm ảnh hưởng đến kết quả toàn bài toán. Với trạng thái ban đầu là t(n) sau khi làm giai đoạn n tốt nhất ta có trạng thái ban đầu của giai đoạn n-1 là t(n-1) và tác động điều khiển của giai đoạn n-1 là d(n-1), có thể tiếp tục xét đến giai đoạn n-1. Sau khi tối ưu giai đoạn n-1 ta lại có t(n-2) và d(n-2) và lại có thể tối ưu giai đoạn n-2 … cho đến khi các giai đoạn từ n giảm đến 1 được tối ưu thì coi như hoàn thành bài toán. Gọi giá trị tối ưu của bài toán tính đến giai đoạn k là Fk giá trị tối ưu của bài toán tính riêng ở giai đoạn k là Gk thì Fk = Fk-1 + Gk
* Bước 2: Tổ chức dữ liệu và chương trình
* Tổ chức dữ liệu sao cho đạt các yêu cầu sau:

+ Dữ liệu được tính toán dần theo các bước.

+ Dữ liệu được lưu trữ để giảm lượng tính toán lặp lại.

+ Kích thước miền nhớ dành cho lưu trữ dữ liệu càng nhỏ càng tốt, kiểu dữ liệu được chọn phù hợp, nên chọn đơn giản dễ truy cập.

* Cụ thể

+ Các giá trị của Fk thường được lưu trữ trong một bảng (mảng một chiều hoặc hai, ba, v.v… chiều).

+ Cần lưu ý khởi trị các giá trị ban đầu của bảng cho thích hợp, đó là các kết quả của các bài toán con có kích cỡ nhỏ nhất của bài toán đang giải.

+ Dựa vào công thức, phương trình truy toán (\*) và các giá trị đã có trong bảng để tìm dần các giá trị còn lại của bảng.

+ Ngoài ra còn cần mảng lưu trữ nghiệm tương ứng với các giá trị tối ưu trong từng gian đoạn.

+ Dựa vào bảng lưu trữ nghiệm và bảng giá trị tối ưu trong từng giai đoạn đã xây dựng, tìm ra kết quả bài toán.

* Bước 3: Tối ưu
* Làm tốt thuật toán bằng cách thu gọn hệ thức (\*) và giảm kích thước miền nhớ. Thường tìm cách dùng mảng một chiều thay cho mảng hai chiều nếu giá trị một dòng (hoặc cột) của mảng hai chiều chỉ phụ thuộc một dòng (hoặc cột) kề trước.
* Trong một số trường hợp có thể thay mảng hai chiều với các giá trị phần tử chỉ nhận giá trị 0, 1 bởi mảng hai chiều mới bằng cách dùng kỹ thuật quản lý bit.

## Các bước giải bài toán quy hoạch động

**Bước 1**: Bắt đầu bằng **các bài toán cơ sở** (thông thường rất dễ) và lưu vào bảng phương án.

**Bước 2**: Dùng công thức truy hồi phối hợp với **lời giải của các bài toán nhỏ đã lưu** trong bảng phương án để tìm lời giải của bài toán lớn hơn và tiếp tục lưu vào bảng phương án.

**Bước 3**: Lặp lại cho đến khi bài toán chính được giải.

**Bước 4**: Từ bảng phương án ta đưa ra được nghiệm tối ưu cho bài toán.

## Các hướng tiếp cận bài toán

### a/ Tabulation Method – Bottom Up

Tabulation Method – Bottom Up: Từ dưới lên, ví dụ ta có n trạng thái và a[i] đại diện cho một trạng thái và a[n] là trạng thái cần tìm, phương pháp giải bottom up sẽ bắt đầu từ những trạng thái đầu tiên a[0], a[1], ... để giải quyết đến a[n]

### b/ Memoization Method – Top Down

Tabulation Method – Bottom Up: Tự trên xuống, để giải được trạng thái a[n] thay vì giải từ trạng thái a[0] như Bottom up, ta tìm lời giải từ các trạng thái có liên quan tới trạng thái cần tìm và giảm đi các trạng thái không cần thiết.

### c/ So sánh 2 cách tiếp cận:

|  |  |  |
| --- | --- | --- |
| **So sánh** | **Bottom up** | **Top down** |
| Các trạng thái | Khó tìm sự liên kết giữa các trạng thái bằng bottom up. | Dễ tìm sự liên kết giữa những trạng thái do chỉ cần tìm những trạng thái liên quan. |
| Thiết kế thuật toán | Đối với những bài toán dài nhiều điều kiện thiết kết sẽ phức tạp. | Thiết kế dễ vì chỉ cần hiểu được những trạng thái liên quan đến trạng thái đích. |
| Tốc độ | Nhanh bởi vì giải quyết trực tiếp vấn đề và các vấn đề con. | Chậm hơn do phải suy ra từ đệ quy thông thường. |
| Giải quyết bài toán con | Dễ hình dung bởi được giải trực tiếp. | Khó hình dung vì phải gián tiếp giải dựa trên những trạng thái khác. |
| Bộ nhớ | Không cần cung cấp liền bộ nhớ để có thể giải được đến cuối bài toán | Bắt buộc phải cung cấp đầy đủ bộ nhớ trước khi giải |

# Các bài toán kinh điển về quy hoạch động

## Bài toán: 0/1 Knapsack

### a/ Bài toán:

Cho một cái cặp chứa được khối lượng giới hạn và một nhóm các đồ vật mà mỗi món bao gồm khối lượng và giá trị riêng của chúng, chọn các món đồ sao cho khi bỏ các món vào cặp, tổng khối lượng của chúng nhỏ hơn hoặc bằng khối lượng giới hạn của cặp và đem lại được tổng giá trị lớn nhất.

### b/ Tóm tắt:

+ Gọi wi và vi lần lượt là khối lượng, giá trị của các món đồ.

+ Gọi W là khối lượng tối đa mà cặp có thể chứa được.

+ x là một chuỗi số 0 hoặc 1, nghĩa là 1 là vật thứ i trong mảng x được thêm vào cặp

### c/ Hướng giải quyết:

1/ Bruteforce:

* Tìm tất cả những chuỗi x đạt giá trị lớn nhất.
* x = {1, 0, ...}, x = {1, 1, ...}, x = {0, 0, ...}, ...
* Độ phức tạp lên tới O()

2/ Dynamic programming – Bottom up:

* Ví dụ chuỗi v = {1, 4, 5, 7}, w = {1, 3, 4, 5}
* Giới hạn W = 7
* Pseudocode để tìm giá trị lớn nhất:

for j from 0 to W do:

m[0, j] := 0

for i from 1 to n do:

for j from 0 to W do:

if w[i] > j then:

m[i, j] := m[i-1, j]

else:

m[i, j] := max(m[i-1, j], m[i-1, j-w[i]] + v[i])

* Độ phức tạp thuật toán: O(n\*W) với n là số vật và W là giới hạn khối lượng.

## Bài toán dãy con đơn điệu dài nhất:

### a/ Bài toán:

Cho dãy a1 ,a2 ,.. an. Hãy tìm độ dài lớn nhất dãy con tăng có nhiều phần tử nhất của dãy.

### b/ Tóm tắt:

Gọi L(i) là độ dài dãy con tăng dài nhất, các phần tử lấy trong miền từ a1 đến ai và phần tử cuối cùng là ai.

### c/ Hướng giải quyết:

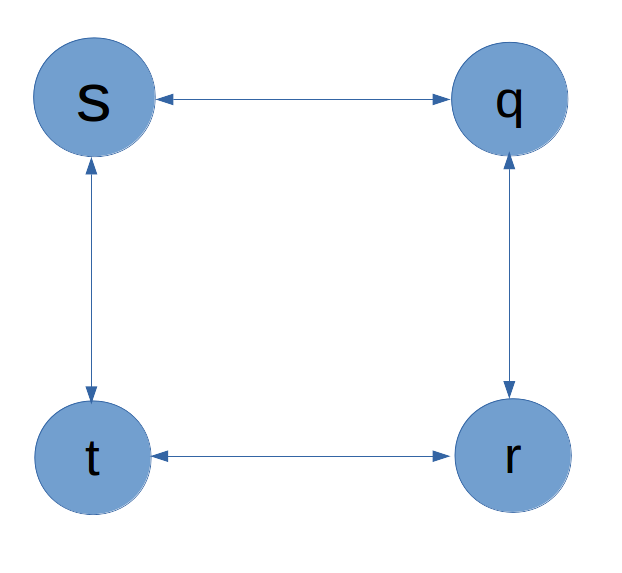
Ta chia 1 bài toán lớn (dãy con của nn số) thành các bài toán con cùng kiểu có kích thước nhỏ hơn (dãy con của dãy i số). Vấn đề là công thức truy hồi để phối hợp kết quả của các bài toán con. Ta có công thức QHĐ để tính Li như sau:

* L1=1.
* Li=max(1,Lj+1) với mọi phần tử jj thỏa mãn: 0<j<i và Aj≤Ai

Tính Li: phần tử đang được xét là Ai. Ta tìm đến phần tử Aj<Ai có Lj lớn nhất. Khi đó nếu bổ sung Ai vào sau dãy con …Aj ta sẽ được dãy con tăng dần dài nhất xét từ A1…Ai.

# Phản ví dụ

**Bài toán:** Tìm đường đi dài nhất từ s sang r mà chỉ đi qua mỗi điểm 1 lần.

****

+ Bài toán có thể chia nhỏ nhưng không có cấu con trúc tối ưu. Khi chia bài toán ra thành nhiều phần, ta có thể chia ra để tìm đường đi từ s sang t và từ t sang r hay từ s sang q và từ q sang

+ Tuy nhiên khi ta giải những vấn đề con,s 🡪 t thì đường đi dài nhất là s 🡪 q 🡪 t 🡪 r, t 🡪 r đường đi dài nhất là t 🡪 s 🡪 q 🡪 r. Kết quả nếu giải theo hướng này sẽ là s 🡪 q 🡪 t 🡪 r 🡪 s 🡪 q 🡪 r dẫn đến kết quả sai. Đường đi dài nhất từ s đến r khi chỉ đi qua mỗi điểm 1 lần là s 🡪 q 🡪 r hay s 🡪 t 🡪 r.

# So sánh giữa quy hoạch động vào chia để trị

|  |  |
| --- | --- |
| **Dynamic programming** | **Divide and Conquer** |
| Chia bài toán thành nhiều phần, những phần đó có thể có cách giải khác nhau. | Chia bài toán ra thành nhiều phần, tuy nhiên những phần đó có cách giải tương tự nhau |
| Chia bài toán thành nhiều phần, những phần đó có thể có cách giải khác nhau. | Các vấn đề lớn phụ thuộc vào kết quả của bài toán con. |
| Nếu tìm ra được kết quả trong một bài toán con thì có thể dừng thuật toán. | Bài toán con chỉ được giải một lần và lưu kết quả, không dùng khi chưa giải được các bài toán con. |
| Đôi khi không dùng những kết quả con để giải bài toán. | Để tìm ra được lời giải bắt buộc phải giải được tối ưu bài toán con. |

# Các ứng dụng của Quy hoạch động

* Ứng dụng trong học sâu: Backpropagation là thuật toán được dùng để tính đạo hàm các hệ số trong neural network với loss function đề rồi áp dụng gradient descent để tìm các hệ số. Cốt lõi của thuật toán là một kiểu tiếp cận bottom-up của quy hoạch động (Lấy kết quả đạo hàm từ node output để tính đạo hàm ở node input).
* Quy hoạch động cũng được ứng dụng rộng rãi trong trí tuệ nhân tạo, đã có nhiều paper (bài báo khoa học) nói về kỹ thuật này.

\* Tài liệu tham khảo:

1. <https://topdev.vn/blog/thuat-toan-quy-hoach-dong/>

2. <https://www.baeldung.com/cs/fibonacci-top-down-vs-bottom-up-dynamic-programming>

3. <https://www.youtube.com/watch?v=vYquumk4nWw>

4. https://www.youtube.com/watch?v=OQ5jsbhAv\_M