Powersets

Δυναμοσύνολα

Νοέμβριος 2019

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Πανεπιστήμιο Ιωαννίνων

Γκόγκος Χρήστος

Powerset – δυναμοσύνολο συνόλου

- Το powerset ενός συνόλου S είναι το σύνολο από όλα τα υποσύνολα του S
- Αν S = {A,B,C} τότε το powerset του S είναι το:
  { {}, {A}, {B}, {C}, {A,B}, {A,C}, {B,C}, {A,B,C} }
- Το πλήθος των στοιχείων του powerset είναι $2^{|S|}$ όπου |S| είναι το πλήθος των στοιχείων του S

Δημιουργία powerset (α’ τρόπος - μέσω δυαδικών αριθμών)

Απαρίθμηση δυαδικών αριθμών

- Χρησιμοποιείται η δυαδική αναπαράσταση κάθε αριθμού στο διάστημα 0 έως $2^{|S|} - 1$
- Η παρουσία/απουσία σε ένα σύνολο του πρώτου στοιχείου του S σηματοδοτείται από το «λιγότερο σημαντικό ψηφίο» του δυαδικού αριθμού

Παράδειγμα με S = {A,B,C,D}:

- 0000 → {}
- 0001 → {A}
- 0010 → {B}
- 0011 → {A,B}
- 0100 → {C}
- 0101 → {A,C}
- 0110 → {B,C}
- 0111 → {A,B,C}
- 1000 → {D}
- 1001 → {A,D}
- 1010 → {B,D}
- 1011 → {A,B,D}
- 1100 → {C,D}
- 1101 → {A,C,D}
- 1111 → {A,B,C,D}

Δημιουργία powerset συνόλου (α’ τρόπος)

```cpp
#include <iostream>

using namespace std;

int main() {
    string items[] = {"A", "B", "C", "D"};
    int n = sizeof items / sizeof items[0];
    int total = 1 << n;
    for (int i = 0; i < total; i++) {
        cout << "{";
        for (int j = 0; j < n; j++) {
            if ((i >> j) & 1)
                cout << items[j];
        }
        cout << "}" << endl;
    }
}
```

```
{}
{A}
{B}
{AB}
{C}
{AC}
{BC}
{ABC}
{D}
{AD}
{BD}
{ABD}
{CD}
{ACD}
{BCD}
{ABCD}
```

powerset1.cpp

Δημιουργία powerset συνόλου (α’ τρόπος – εναλλακτική υλοποίηση με std::bitset)

```cpp
#include <iostream>
#include <bitset>

using namespace std;

int main() {
    string items[] = {"A", "B", "C", "D"};
    const int n = sizeof items / sizeof items[0];
    int total = 1 << n;
    for (int i = 0; i < total; i++) {
        bitset<n> x(i);
        cout << x << "{";
        for (int j = 0; j < n; j++) {
            if (x[j])
                cout << items[j];
        }
        cout << "}" << endl;
    }
}
```

```
{}
{A}
{B}
{AB}
{C}
{AC}
{BC}
{ABC}
{D}
{AD}
{BD}
{ABD}
{CD}
{ACD}
{BCD}
{ABCD}
```

powerset1b.cpp

Δημιουργία powerset συνόλου (β’ τρόπος)

Δημιουργία powerset για το σύνολο S

1. Αρχικοποίηση του powerset ως κενό σύνολο.
2. Για κάθε στοιχείο i του S
   I. Για κάθε σύνολο X που έχει ήδη προστεθεί στο powerset
      A. Δημιουργία ενός νέου συνόλου ως αντίγραφο του X και προσθήκη του i σε αυτό
      B. Προσθήκη του νέου συνόλου στο powerset
   II. Προσθήκη του συνόλου {i} στο powerset
3. Προσθήκη του άδειου συνόλου στο powerset

Παράδειγμα: S = {A,B,C,D}

- {}
- **A:** {{A}}
- **B:** {{A}, {A,B}, {B}}
- **C:** {{A}, {A,B}, {B}, {A,C}, {A,B,C}, {B,C}, {C}}
- **D:** {{A}, {A,B}, {B}, {A,C}, {A,B,C}, {B,C}, {C}, {A,D}, {A,B,D}, {B,D}, {A,C,D}, {A,B,C,D}, {B,C,D}, {C,D}, {D}}
- {{A}, {A,B}, {B}, {A,C}, {A,B,C}, {B,C}, {C}, {A,D}, {A,B,D}, {B,D}, {A,C,D}, {A,B,C,D}, {B,C,D}, {C,D}, {D}, {}}

Δημιουργία powerset συνόλου (β’ τρόπος)

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main() {
    vector<string> items = {"A", "B", "C", "D"};
    vector<vector<string>> powerset;
    for (string item : items) {
        vector<vector<string>> new_sets;
        for (vector<string> a_set : powerset) {
            a_set.push_back(item);
            new_sets.push_back(a_set);
        }
        powerset.insert(powerset.end(), new_sets.begin(), new_sets.end());
        powerset.push_back({item});
    }
    powerset.push_back({});
    for (vector<string> a_set : powerset) {
        cout << "{";
        for_each(a_set.begin(), a_set.end(), [](string item) { cout << item; });
        cout << "}" <<endl;
    }
}
```

```
{A}
{AB}
{B}
{AC}
{ABC}
{BC}
{C}
{AD}
{ABD}
{BD}
{ACD}
{ABCD}
{BCD}
{CD}
{D}
{}
```

powerset2.cpp

Δημιουργία powerset συνόλου (γ' τρόπος-αναδρομικά)

- Για κάθε στοιχείο του αρχικού συνόλου εξετάζονται 2 περιπτώσεις
  - Να αποτελεί μέρος του τρέχοντος υποσυνόλου
  - Να μην αποτελεί μέλος του τρέχοντος υποσυνόλου

- Παράδειγμα με αρχικό σύνολο το {A,B,C,D}:
  - {} → {A}, {}
  - {A}, {} → {A,B}, {A}, {B}, {}
  - {A,B}, {A}, {B}, {} → {A,B,C}, {A,B}, {A,C}, {A}, {B,C}, {B}, {C}, {}
  - {A,B,C}, {A,B}, {A,C}, {A}, {B,C}, {B}, {C}, {} → {A,B,C,D}, {A,B,C} {A,B,D}, {A,B}, {A,C,D}, {A,C}, {A,D}, {A}, {B,C,D}, {B,C}, {B,D}, {B}, {C,D}, {C}, {D}, {}

Δημιουργία powerset συνόλου (γ’ τρόπος)

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
void powerset(vector<string> v, int index, vector<string> curr) {
    int n = v.size();
    if (index == n) {
        cout << "{";
        for_each(curr.begin(), curr.end(), [](string item) { cout << item; });
        cout << "}" << endl;
        return;
    }
    vector<string> new_curr(curr);
    new_curr.push_back(v[index]);
    powerset(v, index+1, new_curr);
    powerset(v, index+1, curr);
}
int main() {
    vector<string> items = {"A", "B", "C", "D"};
    powerset(items, 0, {});
}
```

```
{ABCD}
{ABC}
{ABD}
{AB}
{ACD}
{AC}
{AD}
{A}
{BCD}
{BC}
{BD}
{B}
{CD}
{C}
{D}
{}
```

powerset3.cpp