이산적 선택지가 존재하는 쿠르노 모델(Cournot model)의 최적대응 기반 풀이 기법의 개발: 큐(Queue)와 체이닝(Chaining)을 중심으로

이름 _ 김석준, ksjross@gmail.com

2020학년도 1학기 자료구조

초록

두 player간의 시장경쟁을 연속함수로 묘사한 쿠르노 모델(Cournot model)은 이론적인 상황에서 시장의 원리를 효과적으로 설명하였다. 그러나, 대부분의 상황에서 수요 및 공급에 의한 이윤곡선은 이산적인 상황에서 얻은 불연속적인 값들을 대상으로 나타낸 추세선에 불과하며, 또한 그 값이 자연스럽지 못할 경우 연속함수로 표현하기 어려운 상황이 많이 발생한다. 본 연구에서는 입력된 데이터를 바탕으로 체이닝(Chaining) 방식으로 구성된 연결리스트를 구현한 후, 최적대응을 적용한 선별된 데이터들로 큐(Queue)를 이용해 각 점들에 대해 CCW 알고리즘을 적용하여 내쉬균형을 구하는 등 쿠르노 모델의 균형점을 찾는 풀이 기법을 개발하였다, 이러한 풀이 기법은 제공된 데이터를 효율적으로 선별하여 내쉬 균형점을 신속하게 찾을 수 있었으며, 이에 대한 응용 방향성 또한 넓을 것으로 판단된다.

주제어: 내쉬균형, 최적대응, Cournot model, 큐, 체이닝, CCW 알고리즘

♡ 주요 참고문헌: 윤성우의 열혈 자료구조(윤성우 저), C 로 배우는 쉬운 자료구조(이지영 저), 게임이론: 전략적 사고와 분석의 기초(김광호 저)

♡ 주요 알고리즘: 큐(Queue), 체이닝(Chaining), CCW 알고리즘

Ⅰ. 선행연구 분석의 동기 및 목적

기하 알고리즘은 점, 선, 면 등의 기하 객체들로 이루어진 문제를 해결하는 알고리즘으로, 해석기하, 산술기하 등 기초적인 기하요소부터 시작하여 리만기하, 사교기하, 복소기하에 이르는 난해한 기하까지 다루는 넓은 분야를 다루는 알고리즘이다. 특히 이 기하 알고리즘의 기본이 되는 알고리즘은 평면 위의 세 점의 방향관계를 구하는 알고리즘으로, 일반적으로 CCW(Counter Clockwise) 알고리즘으로 불린다. 본 연구에서 이루어질 이산적 그래프의 분석에서 각 점들은 모두 선분의 형태로 연결 가능하며, 그래프의 교점들을 구하는 과정이 필수적이다. 이에 본 연구자는 현재 존재하는 CCW 알고리즘을 분석하여 이를 이용한 C기반 코드를 제작하고, 더 나아가 이를 본 연구에서 집중적으로 조명될 이산적 쿠르노 모델에 접목시킴으로 효율적인 균형점을 찾기 위한 새로운 알고리즘을 개발하는 것이 목적이다.

Ⅱ. 선행연구의 이론적 배경

1. 벡터(vector)

가. 벡터(vector)

벡터(vector)란 기존의 크기에 대한 정보만을 나타내는 스칼라(scalar)에 반하여, 방향과 크기의 의미를 모두 포함하는 표현 도구이다. 이러한 벡터는 주로 힘, 자기장, 전기장 등의 물리적 개념을 서술할 때 자주 사용되며, 2차원 또는 3차원의 벡터를 주로 사용한다. 벡터는 (a, b), (x, y, z) 형태의 tuple 방식의 표현이 가능하며, 또다른 방식으로는 $\hat{i}, \hat{j}, \hat{k}$ 방식의 단위 벡터식 표현이 있다.

나. 벡터의 외적(cross product)

벡터의 외적(cross product)이란 3차원 유클리드 공간에서 정의된 함수의 일종으로, 어떠한 두 벡터 $\vec{a}$, $\vec{b}$의 외적 연산은 $\vec{a}\times\vec{b}$로써 표현하고 연산의 결과는 벡터로 나타난다. 이때, 그 결과값을 $\vec{c}$라 한다면, $\vec{c}=\vec{a}\times\vec{b}$의 크기는 $|\vec{a}||\vec{b}|\sin\theta$이고 방향은 $\vec{a}$, $\vec{b}$에 모두 수직인 방향이다.

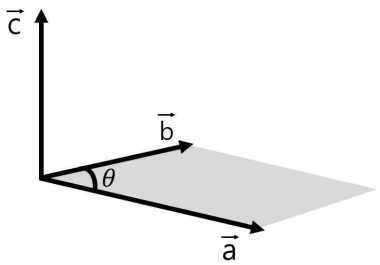


Figure1. 벡터의 외적

다. 벡터의 외적의 연산

일반적으로 벡터의 외적 연산은 정의를 이용한 방법으로 구할 수 있으나, tuple 방식의 성분 형태로 표현될 경우 더 간단한 방법으로 외적 연산이 가능하다.

$\vec{a}=(a_1,a_2,a_3)$, $\vec{b}=(b_1,b_2,b_3)$일 때, $\vec{c}=\vec{a}\times\vec{b}$는 다음과 같다:

$$\vec{c}=(a_2b_3-a_3b_2,\ a_3b_1-a_1b_3,\ a_1b_2-a_2b_1)$$

이는 다음과 같은 행렬로 접근하면 이해가 용이하다.

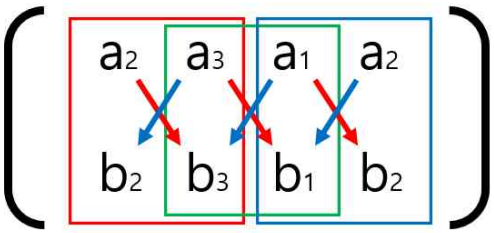


Figure2. 벡터의 외적의 행렬 표현식 접근

2. CCW 알고리즘

가. CCW 알고리즘

CCW(Counter Clockwise) 알고리즘은 2차원 유클리드 공간에 존재하는 임의의 세 점에 대해서, 기준이 되는 두 점에 대해 나머지 점이 어느 위치에 있는지 판별하는 알고리즘이다. 2차원 평면에 존재하는 임의의 세 점 A, B, C에 대해 두 점 A, B를 선별하자. 이후 두 점 A, B를 연결한 벡터 $\overrightarrow{AB}$를 $\vec{a}$라 하고, 기준점 A와 위치 관계를 판별하고자 하는 점 C를 연결한 벡터 $\overrightarrow{AC}$를 $\vec{b}$라 하면, CCW 알고리즘을 적용했을 때 점 C의 위치는 CCW 함수를 통해 다음과 같은 형태로 반환된다.

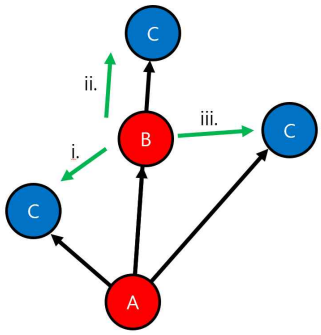


Figure3. CCW 알고리즘

i . $\vec{b}$가 $\vec{a}$에 대해 반시계 방향: 양수
ii . $\vec{b}$가 $\vec{a}$와 같은 방향: 0
iii. $\vec{b}$가 $\vec{a}$에 대해 시계 방향: 음수

이때 사용된 CCW 함수는 두 기준점 A, B와 판별점 C의 좌표정보를 인자로 하는 함수로, CCW(A, B, C)로 나타낸다.

나. CCW 알고리즘을 이용한 선분 교차 여부 판단 알고리즘

이러한 CCW 알고리즘을 응용한 또다른 알고리즘으로는 선분 교차 여부의 판단으로, 네 점의 좌표 정보를 바탕으로 두 선분의 교차 여부를 간단하게 판별하는 알고리즘이다. 이때 임의의 두 선분이 서로에 대해 가질 수 있는 위치관계는 다음과 같은 네가지가 존재한다.

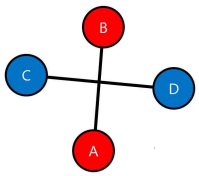
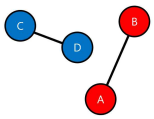
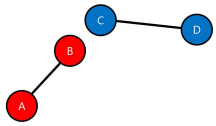
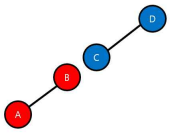
위치 관계	교차	만나지 않음	어느 세 점이 한 직선상에 존재(만나지 않음)	네 점이 한 직선 상에 존재(만나지 않음)
모식도				

table1. 임의의 두 선분이 서로에 대해 가질 수 있는 위치관계

이때 선별해야 하는 경우는 첫 번째 경우인 선분 AB와 선분 CD가 교차되어 있는 상태이다. 이는 기준 벡터 $\overrightarrow{AB}$에 대해서 두 판별 벡터 $\overrightarrow{AC}$, $\overrightarrow{AD}$가 각각 한 벡터가 반시계 방향의 위치에 있을 경우 다른 한 벡터는 시계 방향의 위치에 있다는 뜻으로 해석 가능하다. 따라서 벡터의 외적을 도입하면 기준 벡터 $\overrightarrow{AB}$에 대한 두 판별 벡터 $\overrightarrow{AC}$, $\overrightarrow{AD}$의 외적 결과에서 $\hat{k}$성분의 값이 하나가 양수이면 다른 하나는 음수이므로 그 곱은 음수이다. 단, C, D 중 어느 한 점이 선분 AB 위에 있으면 그 외적 값은 0이 되므로 이 경우에는 0이 된다. 즉 이는 다음과 같이 표현 가능하다: CCW(A, B, C) × CCW(A, B, D) $\leq$ 0

하지만 이 방정식으로만 선분의 교차 여부를 판단해서는 안된다. 이는 세 번째 위치 관계와 중복될 수 있으며 이 경우는 교차가 아니기 때문이다. 따라서, 이 문제는 이번에는 벡터 $\overrightarrow{CD}$를 기준 벡터로 설정하고 두 판별 벡터 $\overrightarrow{CA}$, $\overrightarrow{CB}$에 대해 위의 과정을 반복함으로서 해결 가능하다. 결론적으로 두 선분의 교차를 판단하기 위한 CCW 알고리즘적 판별식은 다음과 같다:

CCW(A, B, C) × CCW(A, B, D) $\leq$ 0 && CCW(C, D, A) × CCW(C, D, B) $\leq$ 0

그러나, 이 외에도 네 번째 경우인 네 점이 모두 한 직선 위에 존재하는 경우를 생각해야 한다. 두 번째 및 세 번째 경우는 모두 위의 판별식으로서 제거되지만, 네 번째 경우는 CCW(A, B, C) × CCW(A, B, D) = 0, CCW(C, D, A) × CCW(C, D, B) = 0을 모두 만족하기 때문에 위의 판별식에 의해 제거되지 않는다. 따라서, CCW(A, B, C) × CCW(A, B, D) == 0 && CCW(C, D, A) × CCW(C, D, B) == 0 일 때에는 서로 다른 선분을 이루는 두 점을 선별하여 상대 위치를 고려해야 한다. 그 방법은 다음과 같다: 우선 네 점이 한 직선 위에 있음을 가정하였으므로 각 점의 x좌표 또는 y좌표 하나만 고려하여도 충분하다. 각 점의 x좌표만을 고려한다고 했을 때, 두 점 A와 B의 x좌표 모두보다 점 C의 x좌표가 더 크거나 작으면 점 C는 선분 AB 위에 있지 않은 점이다. 따라서, 이러한 경우가 발생할시 다음 네 가지 조건을 적용하여 모두 부합하면 이 경우를 제거하도록 알고리즘을 구성한다.

i. (C.x_pos - A.x_pos) × (C.x_pos - B.x_pos) > 0
ii. (D.x_pos - A.x_pos) × (D.x_pos - B.x_pos) > 0
iii. (A.x_pos - C.x_pos) × (A.x_pos - D.x_pos) > 0
iv. (B.x_pos - C.x_pos) × (B.x_pos - D.x_pos) > 0

이 네 가지 조건은 각각 선분 AB 위에 점 C 가 없을 조건, 선분 AB 위에 점 D 가 없을 조건, 선분 CD 위에 점 A 가 없을 조건, 선분 CD 위에 점 B 가 없을 조건을 의미한다.

Ⅲ. 선행연구의 분석

1. CCW 알고리즘

CCW 알고리즘의 기본 원리는 벡터의 외적에 기반하며, 의미 있는 값은 $\overrightarrow{AB}$와 $\overrightarrow{AC}$ 벡터의 외적의 결과 벡터의 $\hat{k}$성분이다. 이는 $a_1b_2 - a_2b_1$으로, 이 성분의 값의 부호를 판별하여 어떤 점 C가 기준 선분 AB에 대해 반시계 방향에 있는지, 시계 방향에 있는지, 일직선 상에 있는지 판단할 수 있다. CCW 알고리즘을 순서도와 C언어로 나타내면 다음과 같다.

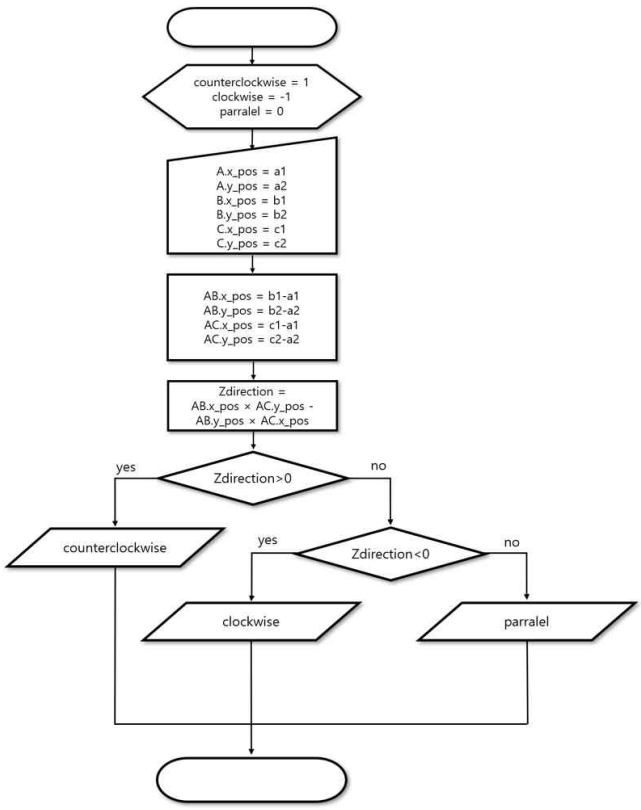


Figure4. CCW 알고리즘의 순서도

```c
#define counterclockwise    1
#define clockwise    -1
#define parallel    0

typedef struct _pos
{
    int x_pos;
    int y_pos;
} Pos;

int CCW(Pos A, Pos B, Pos C)
{
    Pos AB = { B.x_pos - A.x_pos, B.y_pos - A.y_pos };
    Pos AC = { C.x_pos - A.x_pos, C.y_pos - A.y_pos };
    int Zdirection = (AB.x_pos) * (AC.y_pos) - (AB.y_pos) * (AC.x_pos);

    if (Zdirection > 0)
        return counterclockwise;
    else if (Zdirection < 0)
        return clockwise;
    else
        return parallel;
}
```

2. CCW 알고리즘을 이용한 선분 교차 여부 판단 알고리즘

앞에서 보였듯이 CCW 알고리즘을 이용한 선분 교차 여부 판단 알고리즘은 기존에 있는 CCW 알고리즘을 이용하여 임의의 두 선분의 위치 관계를 네 점의 좌표만을 이용하여 구하는 알고리즘이다. 이때 주의해야 할 것은 두 선분이 만나지 않으면서 일직선상에 존재하는 경우로, 이 경우는 따로 생각하여 제거해야 한다. 이 점을 주의하며 CCW 알고리즘을 이용한 선분 교차 여부 판단 알고리즘을 순서도와 C 언어로 나타내면 다음과 같다. 순서도에서 반환값 TRUE 는 두 선분이 교차함을 의미하며, FALSE 는 두 선분이 교차하지 않음을 의미한다.

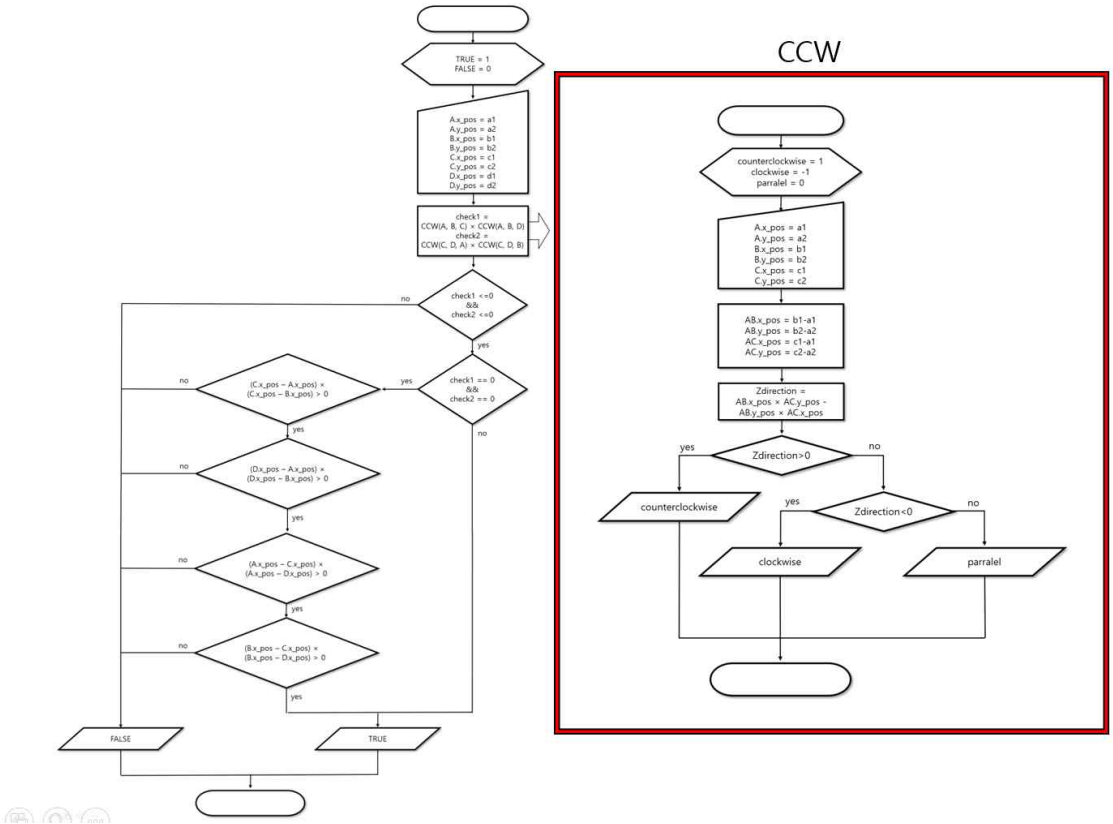


Figure5. CCW 알고리즘을 이용한 선분 교차 여부 판단 알고리즘의 순서도

```
#define counterclockwise    1
#define clockwise    -1
#define parallel    0

#define TRUE    1
#define FALSE    0

typedef struct _pos
{
    int x_pos;
    int y_pos;
} Pos;

int CCW(Pos A, Pos B, Pos C)
{
    Pos AB = { B.x_pos - A.x_pos, B.y_pos - A.y_pos };
    Pos AC = { C.x_pos - A.x_pos, C.y_pos - A.y_pos };
    double Zdirection = (AB.x_pos) * (AC.y_pos) - (AB.y_pos) * (AC.x_pos);

    if (Zdirection > 0)
        return counterclockwise;
```

```
    else if (Zdirection < 0)
        return clockwise;
    else
        return parallel;
}

int CCWbaseCross(Pos A, Pos B, Pos C, Pos D)
{
    int check1 = CCW(A, B, C) * CCW(A, B, D);
    int check2 = CCW(C, D, A) * CCW(C, D, B);

    if (check1 <= 0 && check2 <= 0)
    {
        if (check1 == 0 && check2 == 0)
        {
            if ((C.x_pos - A.x_pos) * (C.x_pos - B.x_pos) > 0)
                if ((D.x_pos - A.x_pos) * (D.x_pos - B.x_pos) > 0)
                    if ((A.x_pos - C.x_pos) * (A.x_pos - D.x_pos) > 0)
                        if ((B.x_pos - C.x_pos) * (B.x_pos - D.x_pos) > 0)
                            return TRUE;
            return FALSE;
        }
        else
            return TRUE;
    }
    else
        return FALSE;
}
```

Ⅳ. 선행연구 고찰 및 결론

위에서 설명한 CCW 알고리즘과 이를 응용한 두 선분의 교차 여부 판별 알고리즘을 활용하여 어떠한 이산적 그래프 상에서 교점을 찾는 연산이 용이할 것으로 판단하였다. 기존의 그래프 교점을 찾는 알고리즘은 대부분 연속함수를 대상으로 이루어진 반면, 위 CCW 알고리즘을 적절히 활용하여 점으로만 이루어진 불연속적 그래프 상에서 교점을 찾을 때 높은 효율을 보일 수 있을 것으로 전망된다.

특히, 2차원 좌표계에서 두 직선의 위치관계를 구하는 데에 주로 사용되던 직선의 방정식과는 달리, 본 CCW 알고리즘은 네 점의 좌표를 복잡한 연산 없이 외적으로만 위치 관계를 파악할 수 있다는 점에서 기울기와 y절편을 모두 구한 뒤 연립해야 하는 직선의 방정식 모델에 비하여 훨씬 효율적이며, 특히 본 연구에서 주로 다루어질 기하 요소는 직선

이 아닌 선분이기 때문에 선분을 연장했을 때 교차되는 경우도 발생할 수 있으므로 이러한 문제점을 배제하기 위한 최적의 알고리즘은 CCW 알고리즘을 입증할 수 있었다.

즉, CCW 알고리즘은 각각의 Nash Equilibrium을 산출해 점의 형식으로 나타내어 쿠르노 모델의 교점을 구하는 본 연구에 있어 매우 적합함을 알 수 있었으며, 이를 이용하여 본 연구의 효율성을 향상시킬 예정이다.

Ⅴ. 본 연구의 동기 및 목적

게임이론(game theory)은 각 경기자(player)의 행위가 다른 player의 행위에 영향을 받는 전략적 상황(strategic situation)에서 행동주체의 의사결정을 연구하는 학문으로, 20세기 초에 시작된 이래 경제학, 생물학, 컴퓨터 공학등 광범위한 분야에서 응용되어지고 있다. 특히, 완비정보(complete information)하의 정태적 게임(static game) 상황을 서술하고 최적의 결론을 이끌어내는 쿠르노 모델은 두 player간의 시장경쟁에서 각 player가 서로의 결정을 예측하고 최적의 생산량을 동시에 결정하는 과점 모형이다. 그러나 게임이론이 다루어지는 대부분의 현실적 상황에서는 player들의 선택이 항상 이산적이며, 그 최적 선택에 대한 그래프 또한 어느 적당한 추세선으로 나타나기 어려운 복잡한 경우가 우세한 반면, 이론적인 쿠르노 모델은 이윤 곡선을 간단한 연속변수로 설정하고 결과값을 도출하는 매우 단순화된 상황만을 묘사하고 있을 뿐이다. 이는 게임이론을 응용할 수 있는 영역을 인위적으로 축소하는 것이며, 보다 현실적인 상황에 적용하여 최적의 결정을 내릴 수 있음에도 불구하고 복잡성의 이유로 간과되어지고 있다는 점에서 연구의 필요성을 나타내고 있다. 이에 본 연구자는 이산적 선택지가 있는 경쟁 구도 속에서 신속하고 정확한 최적 선택을 산출해내는 효율적인 알고리즘 개발의 필요성이 불가피함을 인식하였다. 이러한 쿠르노 모델 분석을 큐(Queue) 자료구조와 체이닝(Chaining) 기법을 적용한 연결 리스트 및 CCW 알고리즘을 도입함으로 구현하고자 하며, 이전 분석의 한계를 빠르고 명확한 계산으로 뛰어넘게 함으로써 실생활 환경에서 쿠르노 모델을 보다 효율적으로 적용할 수 있도록 하고자 한다.

Ⅵ. 본 연구의 이론적 배경 및 방법

1. Nash Equilibrium

가. 보수행렬(payoff matrix)

보수행렬(payoff matrix)은 각 player들의 선택에 따른 보수(payoff)를 명시한 행렬로, 2인의 player들간의 가능한 전략 수가 적을 때 분석하기 유용한 도구이다. 2인의 player간들 간의 게임에서 보수는 다음과 같이 p_1, p_2 형식으로 표현된다. 이때 p_1은 첫 번째 player($player1$)와 두 번째 player($player2$)의 상호간의 선택에 따른 $player1$의 보상, p_2는

둘의 선택에 따른 $player2$의 보상을 의미한다. 보수행렬의 예는 다음과 같다.

		$player2$	
		H	T
$player1$	H	1, 1	4, 0
	T	0, 4	3, 3

table 2. 2인 player 보수행렬 예시

나. Nash Equilibrium

Nash Equilibrium이란 각 player끼리 선택을 했을 때 서로가 자신의 선택을 바꾸지 않는 균형상태를 말한다. $player1$의 임의의 전략 s_1과 $player2$의 임의의 전략 s_2에 대해 그때의 $player1$의 보수를 $u_1(s_1, s_2)$라 하자. 이때, $player1$의 어떤 전략 s_1*이 다음을 만족하면 이 전략 s_1*을 $player1$에게 있어 우월전략이라 정의한다.

s_1*이 $^{\forall}s_1$에 대해 다음을 만족하면 이 전략 s_1*은 $player1$에게 있어 우월전략이다:

$$u_1(s_1^*, s_2) \geq u_1(s_1, s_2)$$

한편, $player1$과 $player2$ 각각의 전략집합 S_1, S_2와 각 집합의 원소중 우월전략 s_1*, s_2*에 대해 Nash Equilibrium은 수학적으로 다음과 같이 정의된다.

2인 게임에서 다음 조건을 만족하면 전략명세(전략쌍) (s_1^*, s_2^*)는 Nash Equilibrium이다:

i . $^{\forall}s_1 \in S_1$에 대해 $u_1(s_1^*, s_2^*) \geq u_1(s_1, s_2^*)$가 성립

ii . $^{\forall}s_2 \in S_2$에 대해 $u_2(s_1^*, s_2^*) \geq u_2(s_1^*, s_2)$가 성립

다. 최적대응(best response)을 이용한 Nash Equilibrium의 풀이

Nash Equilibrium을 해결하기 위한 방법으로 제시되는 풀이 중 하나는 최적대응(best response)을 이용하는 것이다. 이는 특정 player의 선택을 상수로 고정하고 그 선택에 따른 상대 player의 우월전략을 판단하는 과정을 반복한 뒤, 결과적으로 두 player 모두에게 우월전략이 되는 전략명세가 Nash Equilibrium이라는 풀이이다. 최적대응은 수학적으로 다음과 같이 정의된다.

주어진 player2의 전략 s_2에 대한 player1의 최적대응 $BR_1(s_2)$는 $u_1(s_1, s_2)$를 극대화하는 player1의 전략 s_1을 가리킨다. 즉, $BR_1(s_2) \in S_1$은 다음을 만족하는 전략이다:

$^{\forall}s_1 \in S_1$에 대해 $u_1(BR_1(s_2), s_2) \geq u_1(s_1, s_2)$가 성립

위 table2에 표현된 보수행렬을 바탕으로 최적대응법을 적용하여 Nash Equilibrium을 만족하는 전략명세를 찾으면 다음과 같이 (H, H)가 됨을 보일 수 있다.

		*player*2	
		H	T
*player*1	H	1, 1	4, 0
	T	0, 4	3, 3

table 3. 2인 player 보수행렬의 Nash Equilibrium 풀이

2. 쿠르노 모델(Cournot model)

가. 쿠르노 모델

쿠르노 모델은 두 기업이 경쟁하는 시장인 복점(duopoly)에서의 기업 간 경쟁 양상을 해석하기 위해 제시된 모형이다. 이는 Nash Equilibrium이 개발되기 이전의 이론이나 그 풀이의 핵심은 Nash Equilibrium과 동일하며, 쿠르노 모델은 시장의 재화를 연속변수로 설정하고, 두 기업의 경쟁 결과 두 기업의 생산량의 균형은 어떻게 맞추어질지 예측하는 모형이라 볼 수 있다.

나. 쿠르노 모델의 해법

두 기업(player) 1과 2가 동일한 질의 재화를 생산할 때, 각 기업이 직면하는 시장역수요 P는 시장 공급량 Q와 상수 a에 대해 $P=a-Q$로 나타난다. 두 기업의 생산량을 q_1, q_2라 하면 $Q=q_1+q_2$이다. 이때, 각 기업이 제화를 생산할 때 필요한 비용에는 고정비용과 가변비용이 있는데, 고정비용은 생산량에 관계 없이 일정하게 드는 비용이고 가변비용은 생산량에 의존하는 비용이다. 이 가변비용 함수를 미분한 증가분을 한계비용이라 한다. 기본적인 쿠르노 모형에서는 고정비용은 없고 한계비용이 상수 c로 일정한 직선형 비용함수를 채택한다. 따라서, 비용함수는 $C(q)=cq$로 표현된다. 이러한 상황에서 두 기업은 독립적으로 생산량을 결정하고, 그에 따른 총생산량이 결정되면 시장수요에 의해 시장가격이 결정되며, 각 기업의 판매수입 및 이윤이 책정되는 양상을 보이게 된다. 여기서 $a>c$를 가정하면, 주어진 q_2에 대해 기업 1이 q_1을 생산할 경우의 이윤은 $G(q_1,q_2)=Pq_1-cq_1=\{a-(q_1+q_2)\}q_1-cq_1=\{a-c-(q_1+q_2)\}q_1$이므로 기업 1은 다음과 같은 최적화 문제를 풀게 된다: $\max_{q_1}G(q_1,q_2)=\max_{q_1}\{a-c-(q_1+q_2)\}q_1$

이때 두 변수 q_1, q_2가 식에 존재하는데, 이중 q_1에 대한 최대 이윤을 따져야하므로 주어진 식을 q_1에 대해 편미분하면 다음과 같다.

$$\frac{\partial}{\partial q_1}G(q_1,q_2)=\frac{\partial}{\partial q_1}\{a-c-(q_1+q_2)\}q_1=-q_1+a-c-(q_1+q_2)=a-c-q_2-2q_1=0$$

$$\therefore q_1=\frac{a-c-q_2}{2}=-\frac{1}{2}q_2+\frac{a-c}{2}=BR_1(q_2)$$

$$BR_1(q_2)=-\frac{1}{2}q_2+\frac{a-c}{2}$$

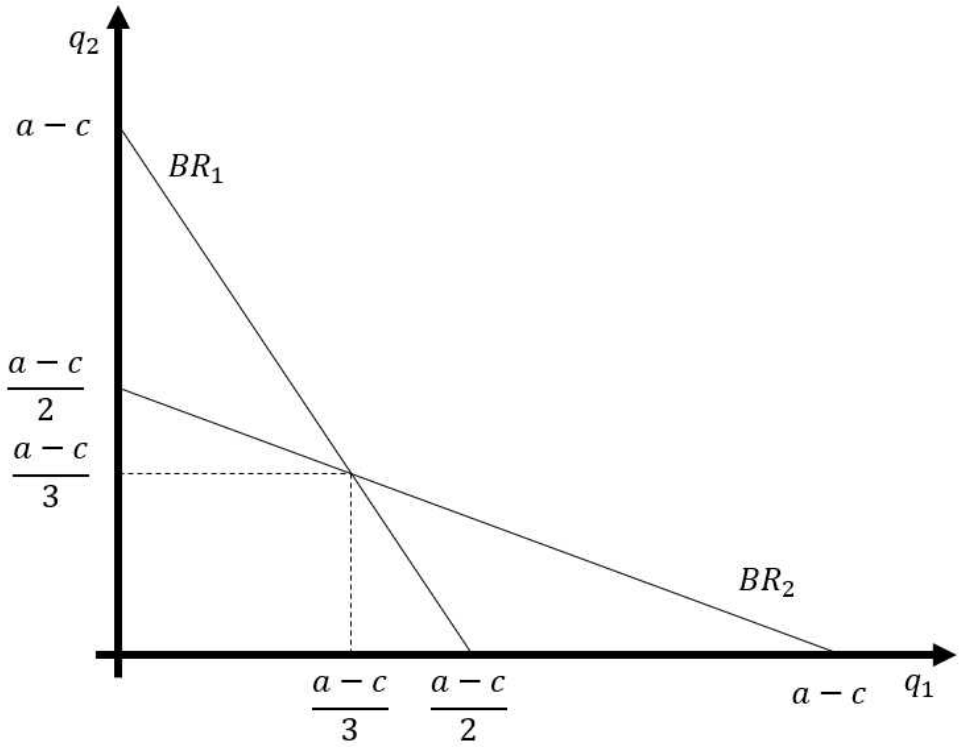
마찬가지 방식으로 기업 2의 최적대응을 구하면 다음과 같다.

$$BR_2(q_1) = \frac{a-c-q_1}{2} = -\frac{1}{2}q_1 + \frac{a-c}{2}$$

두 최적대응식을 연립하여 풀면 다음과 같은 Nash Equilibrium을 얻는다.

$$q_1^* = q_2^* = \frac{a-c}{3}$$

따라서, 서로가 $\frac{a-c}{3}$만큼의 생산을 할 때 서로에게 최적이 된다. 이를 그래프로 나타내면 다음과 같다.



graph1. 쿠르노 모델의 그래프적 해석

3. 큐(Queue)

가. 큐(Queue)

큐(Queue) 자료구조는 선입선출, FIFO(First-In, First-Out) 구조의 자료구조이다. 큐 자료구조의 ADT는 정형화된 편에 속하며, 큐의 ADT에는 다음이 있다.

```
ADT Queue

  QueueInit(Q) ::= create empty Queue;
  QIsEmpty(Q) ::= if Q is empty then return TRUE(1);
                  else return FALSE(0);
  Enqueue(Q, data) ::= insert data at the rear or Q;
  Dequeue(Q) ::= if (QIsEmpty(Q)) then return error;
                  else remove and return the front element of Q;
  Peek(Q) ::= if (QIsEmpty) then return error;
               else return the front element of Q;

End Queue
```

본 연구에서는 이러한 큐의 정의를 이용하여 많은 양의 전략명세 data를 수용하여 이를 빠르게 정리해 최대의 이익을 얻을 수 있는 선택을 결정하게 하고자 한다.

4. 체이닝(Chaining)

가. 체이닝(Chaining)

체이닝(Chaining) 기법은 탐색 알고리즘에서 해쉬(Hash)를 이용한 데이터의 탐색 과정에서 나타나는 충돌(collision)을 해결하는 방법 중 하나이다. 일반적으로 체이닝은 연결리스트를 이용해 구현하며, 체이닝 기반의 테이블을 도식화하면 다음과 같다.

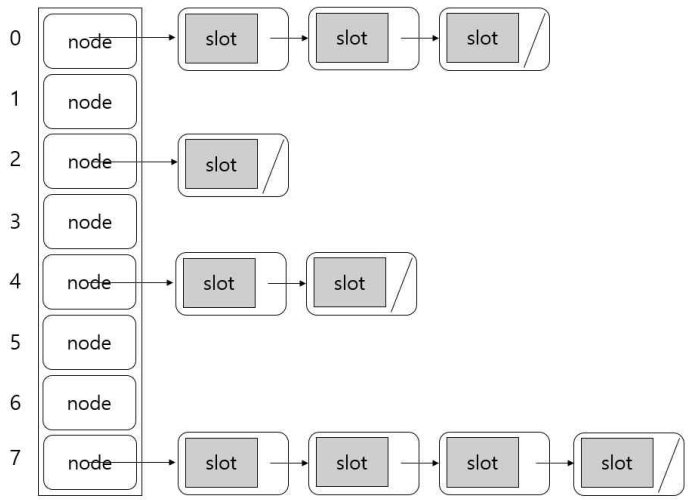


Figure6. 체이닝(Chaining)

Ⅶ. 본 연구 주요 알고리즘 및 프로그램 분석

1. 체이닝(Chaining)기법과 큐(Queue), CCW를 이용한 쿠르노 모델(Cournot model)의 해법에 대한 알고리즘

가. 쿠르노 모델(Cournot model)의 해법에 대한 알고리즘

본 연구에서는 체이닝(Chaining), 큐(Queue), CCW 알고리즘을 이용하여 쿠르노 모델(Cournot model)에 대한 해를 구하는 알고리즘을 제시한다.

먼저, 체이닝을 활용한 전략 명세들로 이루어진 체이닝 연결 리스트에 데이터를 저장한다. 이때, 입력값은 player1과 player2에 따라 다른데, 이 때문에 체이닝 연결 리스트는 총 2개가 필요하다. 한 player이 기준인 데이터를 입력하는 경우 이는 다른 player의 생산량(product_other)에 종속된 데이터를 입력하는 것이다. 따라서, 이 경우 다른 player의 생산량(product_other)와 그 때의 자신의 생산량(product_mine) 및 그러한 전략이 생겼을 경우 자신에게 오는 이윤(profit_mine)으로 구성된 총 3개의 인자를 받아 체이닝 리스트에 삽입한다. 이때, 상대 생산량이 동일한 상황에서 자신의 생산량 또는 이윤이 이미 입력된 값과 중복되는 값이 존재할 경우 이는 처리가 불가능하므로 데이터를 삽입하지 않는다.

이후 쿠르노 모델의 해를 찾고자 할 때에는 큐를 이용해 최적대응에 의해 선별한 전략명세를 저장한다. 큐에 삽입되기 전에는 상대 product(product_other)를 기준으로 오름차순으로 listNode를 정렬한다. 정렬이 완료되면 처음부터 listNode를 하나씩 방문하며 연결된 체인에서 자신의 이윤(profit_mine)이 가장 높게 될 때의 자신의 생산량(product_mine)과 방문중인 listNode의 product(product_other)를 함께 묶어 큐(Queue)에 삽입한다. 모든 listNode를 순차적으로 방문한 이후에는 완성된 큐(Queue)를 반환한다. player1과 player2에 의해 총 2개의 체이닝 리스트가 존재하므로 이러한 큐는 총 2개가 형성된다.

마지막으로 생성된 큐(Queue)를 이용하여 순차적으로 큐(Queue)에서 데이터를 2개씩 참조해 교점을 구한다. 이때, 각 전략명세는 순차적으로 쌍을 맺어야 하므로 2개의 데이터를 참조할 때 하나는 Enqueue 연산으로, 하나는 Peek 연산으로 한다. 만약 CCW 알고리즘에 의해 교점이 발생한다면 이를 출력한다.

쿠르노 모델의 해를 찾는 실행도는 오른쪽과 같다.

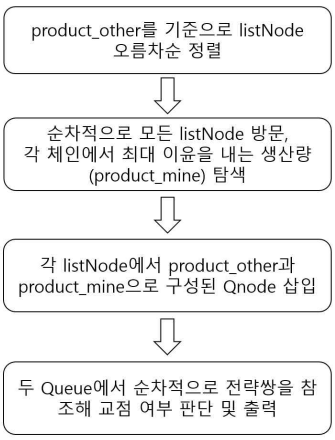


Figure7. 쿠르노 모델(Cournot model) 해 탐색 실행도

2. 쿠르노 모델(Cournot model)의 해법에 대한 함수

가. 체이닝(Chaining)을 활용한 전략 명세(Strategy) 데이터의 관리

본 연구에서는 체이닝(Chaining) 기반 연결리스트에 사용자가 전략명세 데이터를 입력하고 이에 대해 최적대응에 해당되는 순서쌍을 큐(Queue)에 삽입하여 Nash Equilibrium을 구하는 알고리즘을 제시한다. 입력되는 정보는 대상이 되는 player에 따라 두 개의 체이닝 리스트로 구분되어 저장되며, 체이닝 리스트를 구성하는 각 리스트 노드(listNode)는 슬롯 노드(slotNode)들이 연결 리스트 형태로 구현된 체인(chain)을 포함한다. 이를 도식화 하면 다음과 같다.

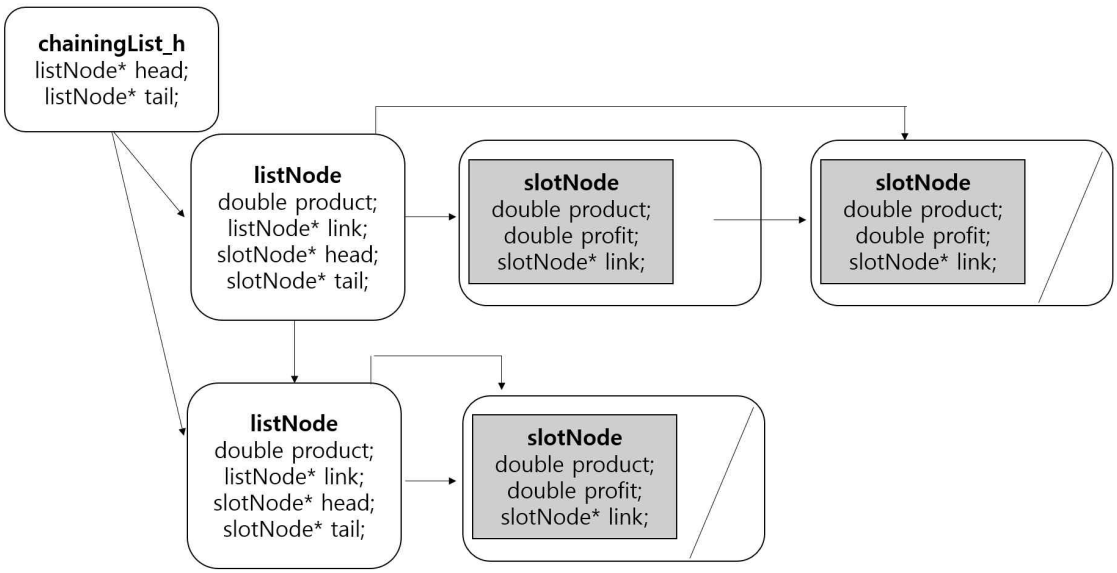


Figure8. 체이닝(Chaining) 기반 연결리스트

이러한 체이닝 기반 연결리스트를 생성하고 이에 데이터를 삽입 및 삭제하는 연산을 중심으로 체이닝 기반 연결리스트 함수를 설계하였다. ChainingBasedLinkedList.c에서 제작한 체이닝 기반 연결리스트와 관련된 주요 함수는 다음과 같다.

```
createChainingList_h() ::= create empty chainingList;
isPossible(L, product_other, product_mine, profit_mine) ::=
                    if valid data then return TRUE(1);
                    else return FALSE(0);
insertSlotNode(L, product_other, product_mine, profit_mine) ::=
                    if isPossible() then insert new slot in chaining list L;
                      return TRUE(1);
                    else return FALSE(0);
deleteSlotNode(L, product_other, product_mine) ::=
                    remove the element of L;
```

이때 isPossible 함수에서의 적절한(valid) 데이터는 체이닝 리스트 L에

product_other가 존재할 때, 이에 연결된 체인(chain)에 product_mine과 동일한 값의 product가 존재하지 않으며 profit_mine과 동일한 값의 profit이 존재하지 않는 것이다.

나. 큐(Queue)를 이용한 최적대응 데이터의 참조

본 연구에서는 보다 일반적인 ChainingBasedLinkedList.c를 연구에 적합한 형태로 특수화한 Cournot.c를 사용하였다. Cournot.c에서는 저장된 각 데이터들을 대상으로 최적대응을 적용하여 선정한 전략쌍(Strategy)를 노드로 갖는 큐(Queue)를 구현하여 이를 참조하는 방식으로 Nash Equilibrium을 구하는 방법을 모색하였다. 이때 전략쌍은 사전에 정렬하여 크기에 맞도록 배치되며, 정렬된 전략쌍을 순서대로 큐에 삽입한 뒤 참조하여 교점을 찾는다. Cournot.c에 정의된 주요 함수들은 다음과 같다.

```
createPayoff() ::= create empty payoff;
insertNewStrategy(L, product_other, product_mine, profit_mine) ::=
                    if isPossible() then insert new strategy in payoff L
                      and return TRUE(1);
                    else return FALSE(0);
deleteStrategy(L, product_other, product_mine) ::=
                    remove the element of L;
SortListNode(L) ::= sort listNode based on product_other;
ShowCross(L1, L2) ::= print intersection range based on CCW;
ShowPayoff(L) ::= print all strategy;
```

다. CCW 알고리즘을 이용한 Nash Equilibrium 탐색

큐(Queue)를 이용해 저장한 정렬된 전략쌍은 ShowCross에 의해 교점 형성 여부가 판단된다. 이때 CCW.c의 함수를 기준으로 판단하며, 주요 함수들은 다음과 같다.

```
CCW(A, B, C) ::= return sign of cross product AB, AC;
CCWbaseCross(A, B, C, D) ::= if AB and CD intersected then return
                               TRUE(1)
                               else return FALSE(0)
```

3. 쿠르노 모델(Cournot model)의 해법에 대한 기능과 실행 결과

가. 쿠르노 모델(Cournot model) 해법에 대한 기능

본 연구에서는 위에서 제시한 함수를 바탕으로 쿠르노 모델(Cournot model)의 해법을 유연하게 제시할 수 있는 기능을 제시한다. 연결리스트를 기반으로 이루어지므로 메모리의 동적 할당 및 삭제가 용이하며, 이를 이용하여 사용자의 입력 데이터를 바탕으로 교점의 범위를 효율적으로 제시하는 기능을 제공한다. 다음은 본 프로그램에 기능에 대한 소개를 축약하여 나타낸 것이다.

1. 새로운 전략(Strategy) 입력
-player2에 대한 player1의 전략 입력
-player1에 대한 player2의 전략 입력
2. 전략(Strategy) 삭제
-player2에 대한 player1의 전략 삭제
-player1에 대한 player2의 전략 삭제
3. 모든 전략(Strategy) 출력
4. 내쉬균형지점(Nash Equilibrium) 출력
5. 종료

나. 쿠르노 모델(Cournot model) 해법에 대한 실행 결과

본 프로젝트에서 정의한 함수 및 알고리즘을 바탕으로 임의의 점들을 입력하여 교점을 출력하는 실행 결과를 나타낸 화면이다.

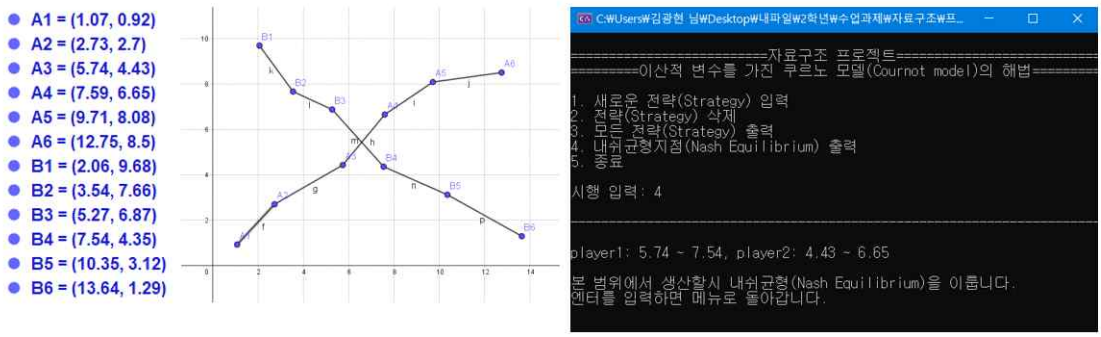


Figure9. 쿠르노 모델(Cournot model) 해법 프로그램 실행 결과

Ⅷ. 본 연구의 결과

본 연구에서는 사용자가 입력한 데이터를 바탕으로 내쉬 균형(Nash Equilibrium)을 출력하는 결과를 보였다. 이를 통해 단순한 이산적 값들을 바탕으로 빠르게 내쉬 균형(Nash Equilibrium)을 구할 수 있음을 연구를 통해 확인할 수 있었다.

Ⅸ. 본 연구의 고찰 및 결론

본 연구를 통해 수가 적거나 추세선에 맞지 않아 연속 함수로 표기할 수 없거나 연속 함수를 만들어도 교점을 구하는 과정이 복잡하고 시간이 오래 걸린다는 기존 쿠르노 모델(Cournot model)의 해법을 체이닝(Chaining), 큐(Queue) 및 CCW 알고리즘과 같은 간단하고 효율적인 자료구조와 알고리즘을 활용하여 구하는 새로운 방식을 제시하였다. 이를 통해 이산적 데이터를 바탕으로 쉽고 빠르게 내쉬 균형점(Nash Equilibrium)을 찾을

수 있는 본 연구로써 향상된 시장 분석 및 기타 연구 등에서 널리 활용될 수 있을 것으로 기대된다.

X. 참고문헌

1. 김광호. (2018). 게임이론: 전략적 사고와 분석의 기초. 시그마프레스
2. 자손9319. CCW와 CCW를 이용한 선분 교차 판별. https://jason9319.tistory.com/358
3. 윤성우. (2012). 윤성우의 열혈 자료구조. Orange Media
4. 이지영. (2016). C로 배우는 쉬운 자료구조. 한빛아카데미
5. 위키백과. 큐 (자료구조). https://ko.wikipedia.org/wiki/%ED%81%90_(%EC%9E%90%EB%A3%8C_%EA%B5%AC%EC%A1%B0)